Consommation d'énergie dans les interconnexions sur puce : Estimation de haut niveau et optimisations architecturales

Antoine Courtay

N^o d'ordre :

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE BRETAGNE SUD

pour obtenir

le grade de : *DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE SUD*

Mention : Electronique et Informatique Industrielle

par

Antoine COURTAY

Équipes d'accueil :	Lab-STICC - Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance - Lorient IRISA - Institut de Recherche en Informatique et Systèmes Aléatoires - Lannion
École Doctorale :	Santé, Information, Communication, Matière et Mathématiques
Composante Universitaire :	UFR Sciences et Sciences pour l'Ingénieur

Consommation d'énergie dans les interconnexions sur puce : Estimation de haut niveau et optimisations architecturales

SOUTENUE LE 25 novembre 2008 devant la commission d'examen

COMPOSITION DU JURY :

M.	BELLEVILLE	CEA LETI	Rapporteur
N.	JULIEN	Université de Bretagne Sud	Directrice de thèse
Y.	LEDUC	Texas Instrument	
S.	PIESTRAK	Université de Metz	
B.	ROUZEYRE	Université de Montpellier 2	Rapporteur
O.	SENTIEYS	Université de Rennes 1	Directeur de thèse

N^o d'ordre :

THÈSE

présentée

DEVANT L'UNIVERSITÉ DE BRETAGNE SUD

pour obtenir

le grade de : *DOCTEUR DE L'UNIVERSITÉ DE BRETAGNE SUD*

Mention : Electronique et Informatique Industrielle

par

Antoine COURTAY

Équipes d'accueil :	Lab-STICC - Laboratoire des Sciences et Techniques de l'Information, de la Communication et de la Connaissance - Lorient IRISA - Institut de Recherche en Informatique et Systèmes Aléatoires - Lannion
École Doctorale :	Santé, Information, Communication, Matière et Mathématiques
Composante Universitaire :	UFR Sciences et Sciences pour l'Ingénieur

Consommation d'énergie dans les interconnexions sur puce : Estimation de haut niveau et optimisations architecturales

SOUTENUE LE 25 novembre 2008 devant la commission d'examen

COMPOSITION DU JURY :

M.	BELLEVILLE	CEA LETI	Rapporteur
N.	JULIEN	Université de Bretagne Sud	Directrice de thèse
Y.	LEDUC	Texas Instrument	
S.	PIESTRAK	Université de Metz	
B.	ROUZEYRE	Université de Montpellier 2	Rapporteur
O.	SENTIEYS	Université de Rennes 1	Directeur de thèse

Remerciements

Cette thèse s'étant effectuée en collaboration avec le laboratoire *Lab-STICC* à Lorient et l'*IRISA* à Lannion, je remercie tout d'abord Emmanuel Boutillon et Marc Sevaux, Professeurs des Universités et Directeur du laboratoire *Lab-STICC* à mon entrée et à ma sortie de thèse ainsi que Olivier Sentieys, Professeur des Universités et responsable de l'équipe *CAIRN* pour m'avoir accueilli dans leurs équipes de recherche respectives.

Je tiens ensuite à remercier Nathalie Julien et Olivier Sentieys, Professeurs des Universités, respectivement ma directrice et mon co-directeur de thèse, de m'avoir fait confiance en me permettant d'effectuer ma thèse au sein des équipes *Low-Power* au *Lab-STICC* et *CAIRN* à l'*IRISA*.

Je remercie tout particulièrement Johann Laurent, Maître de Conférence, mon encadrant de thèse, pour sa disponibilité, ses précieux conseils qu'il m'a apportés tout au long de ces trois années.

Je remercie Bruno Rouzeyre, Professeurs des Universités, pour avoir accepté d'être rapporteur de ce travail.

Je remercie Marc Belleville, membre du *CEA LETI*, pour avoir accepté d'être rapporteur de ce travail.

Je tiens également à remercier Stanislaw Piestrak et Yves Leduc, respectivement Professeurs des Universités et Ingénieur chez *Texas Instrument* pour leur participation à ce jury.

Je remercie toutes les personnes, membres des mes deux laboratoires d'accueil que j'ai cotoyées au cours de ces trois dernières années pour leur amitié et leur bonne humeur.

Enfin, je remercie ma famille. C'est également grace à leur soutien que cette thèse a pu être menée.

Sommaire

Introduction

Problématique

Avec l'évolution du nombre de transistors par puce, les systèmes électroniques actuels permettent de réaliser de plus en plus de fonctionnalités. Cette augmentation grandissante de l'intégration permet actuellement de fabriquer des circuits comportant plusieurs centaines de millions de transistors tel que le montre la figure 1. Cette évolution (globalement exponentielle), correspond à la loi de variation proposée par Moore dans [Moo65]. Un système électronique actuel ne se contente plus uniquement de réaliser une seule fonctionnalité dédiée à une application mais devient pluri-applicatif. En effet, ce n'est plus un fragment d'application que le circuit va réaliser mais un système dans son ensemble : c'est ce que l'on nomme couramment Système sur puce (System on Chip : SoC).

Un SoC est généralement composé de plusieurs éléments tels que des coeurs de processeurs, des accélérateurs matériels et de la mémoire par exemple. Afin de faire communiquer tous ces éléments, un réseau d'interconnexion très dense est également nécessaire. Les interconnexions dans un SoC sont actuellement devenues un des goulots d'étranglement ; en effet, les différents éléments constituant le système peuvent très bien avoir des performances temporelles remarquables, si le réseau d'interconnexion ne suit pas la cadence, c'est toute l'intégrité du système qui se voit menacée.

A l'heure actuelle, le critère de consommation est devenu un enjeu primordial dans la conception d'un système, où, il n'y a encore que quelques années, seules les performances et la surface occupée sur la puce étaient des paramètres critiques. En effet, l'évolution de la capacité des batteries des systèmes portables ne suit pas l'évolution des performances de ces mêmes systèmes. Or, la demande de performance étant un critère de plus en plus recherché par les utilisateurs, nous voyons apparaître un nouveau compromis performance/autonomie. De nombreuses techniques ont été proposées afin de réduire la consommation des différents éléments entrant dans la composition d'un système. Depuis plusieurs années, les concepteurs sont contraints d'optimiser leur système tant d'un point de vue temporel que consommation. Jusqu'à récement, les interconnexions n'étaient optimisées que temporellement et toutes les optimisations de consommation se focalisaient sur la logique. Mais aujourd'hui, du fait de la part grandissante des interconnexions dans le budget consommation d'un système (plus de 50% de la puissance dynamique à l'heure actuelle), l'effort ne doit plus être porté uniquement sur la logique de calcul.

Un système se doit également d'être fiable ; en effet, dans les technologies actuelles, a cause de la réduction des dimensions technologiques, nombre de phénomènes capacitifs internes (autrefois

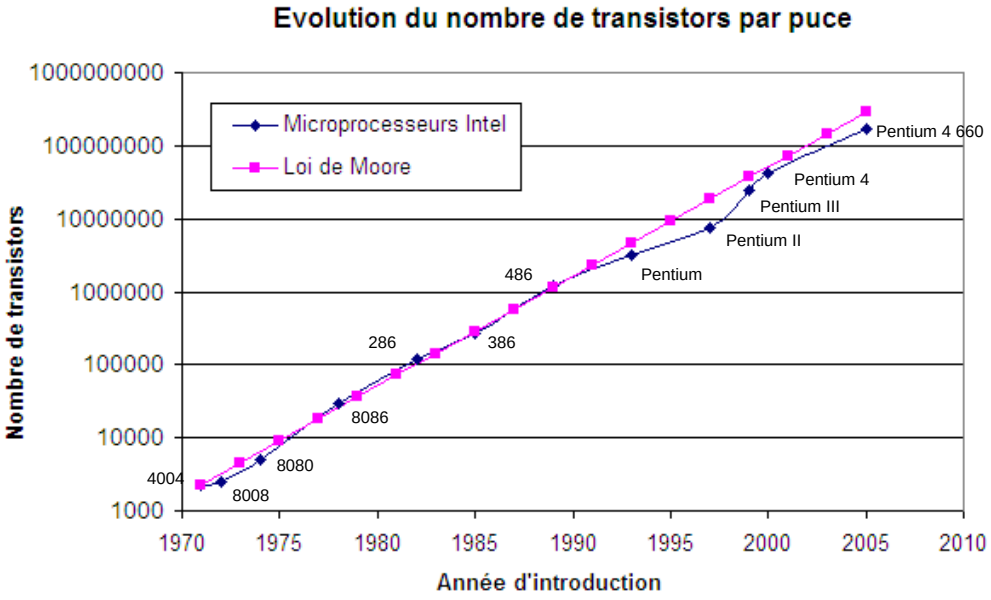


FIG. 1 – Evolution du nombre de transistors par puce pour les microprocesseurs de la famille Intel par rapport aux prédictions faites dans [Moo65].

négligeables) viennent perturber le fonctionnement des circuits et tout particulièrement les interconnexions. Par ailleurs, il existe également beaucoup de bruits externes à la puce (champs magnétiques, orage, lignes haute tension, émetteurs radio, téléphones mobiles et four micro-ondes entres autres) qui peuvent perturber son fonctionnement. Nous nous concentrerons ici sur les phénomènes internes à la puce et reviendrons en détail sur les phénomènes de couplages capacitifs dans la suite du mémoire.

Contributions du mémoire

Nous venons de voir que les interconnexions représentent depuis peu un des points majeurs de la conception d'un système, ceci étant dû au fait qu'elles sont de plus en plus nombreuses, qu'elles posent un problème de délai (délai d'un fil supérieur à celui d'une porte), qu'elles sont également source de consommation importante et sensibles au bruit. Il devient donc indispensable de prendre en compte les interconnexions lors des premières phases de la conception d'une puce. Pour cela, des modèles précis des interconnexions doivent être proposés, c'est à dire des modèles au niveau physique prenant en compte les phénomènes de diaphonie (*crosstalk*). Des outils d'estimation (issus de la simulation des modèles physiques) doivent également être proposés, afin de fournir aux concepteurs des retours rapides et fiables sur leur *design*. Enfin, il est nécessaire d'élaborer des techniques d'optimisation et de quantifier leur impact par le biais entre autre d'outils d'estimation.

Organisation du document

Ce mémoire est organisé en quatre chapitres.
Le premier chapitre de la thèse se propose d'aborder la modélisation de la consommation d'un bus à l'aide de modèles physiques des différents éléments entrant dans sa composition. Le fil, sous forme de modèles résistifs et capacitifs distribués, a d'abord été caractérisé. Puis, au niveau bus, nous avons caractérisé les buffers ainsi que les diaphonies capacitives entre fils.
Dans le second chapitre, la méthode d'estimation de la consommation des interconnexions est proposée. Suite à la modélisation du bus au niveau technologique, les paramètres importants intervenant dans la variation de la consommation (technologie, couche de métal, longueur de bus...) ont été extraits. Des simulations SPICE de ces circuits ont été réalisées ; les résultats expérimentaux ont permis d'obtenir des modèles inclus au sein d'un outil d'estimation. Cet outil (*Interconnect Explorer*) permet alors à l'utilisateur, après configuration, (c'est-à-dire choix de la technologie, de la couche de métal, de la longueur de bus) d'obtenir très rapidement une estimation de la consommation du transfert de données sur un bus. Les expérimentations de validation montrent que l'outil permet d'obtenir une estimation avec une erreur maximale de 3% (par rapport aux simulations SPICE) avec un temps d'exécution de quelques secondes (une simulation SPICE dans les mêmes conditions expérimentales prenant plusieurs heures).
Dans le troisième chapitre, un état de l'art des principales techniques d'optimisation de la consommation et du délai est présenté. L'outil d'estimation présenté dans le chapitre précédent nous permet de valider l'efficacité de ces techniques sur les paramètres impactant la consommation (activité, temps de propagation, capacités parasites...). Dans un second temps, l'analyse des résultats fournis par l'outil permet de montrer que les techniques d'optimisation n'agissent pas forcément sur les bons paramètres. A la fin de ce chapitre, de nouvelles pistes d'optimisation, en adéquation avec les résultats précédents, sont proposées.
Le quatrième chapitre présente les techniques d'optimisation au niveau architectural auxquelles nous avons abouties en se basant sur les pistes d'optimisation du chapitre précédent. Ces techniques (dont une est brevetée : *Spatial Switching*) ont pour particularité de nécessiter un surcoût matériel relativement faible. En effet, nombre de méthodes présentées dans la littérature ont un surcoût matériel assez important, en particulier dû aux codeurs et décodeurs (codecs). Ces codecs engendrent un surcoût en consommation bien souvent supérieur à la réduction apportée sur le bus pour des longueurs d'interconnexions usuelles dans les *SoC* actuels. Nos résultats expérimentaux sur le *Spatial Switching* montrent des gains en consommation pouvant atteindre une réduction de 13% de consommation d'énergie pour un bus de 5mm en 65nm. Ces résultats incluent bien évidemment la consommation due aux codecs. Les gains augmentent encore avec les sauts technologiques ainsi qu'avec l'augmentation de la longueur du bus.
Nous proposerons également une extension possible de nos travaux (outil et modèles) par l'élévation du niveau d'abstraction. En effet, dans ce mémoire, les interconnexions point à point sont notre principale préoccupation ; or, les systèmes actuels peuvent utiliser des réseaux de communication plus complexes. Dans un premier temps, notre approche peut être utilisée pour modéliser des interconnexions de type *MESH* ou *NoC* souvent utilisées dans le cadre de systèmes *MPSoC* (utilisation des résultats de la plate forme SocLib). Dans un second temps, ces résultats et les

précédents peuvent être étendus afin d'être utilisés dans une approche *MDE* (*Model Driven Engineering*). Dans ce cadre, nos travaux s'intégreront dans le projet *ITEA SPICES* qui utilise un profil *AADL* (*Application & Architecture Design Language*), le but étant, ici, d'intégrer nos résultats dans le "*framework*" *OSATE* afin de pouvoir estimer la consommation des communications dès les premières phases de conception.
La consommation des interconnexions étant devenue un enjeu majeur dans la conception de systèmes, nous concluons la thèse par une présentation des futures technologies d'interconnexions alternatives à la conception classique : interconnexions optiques, *SoC* 3D, nanotubes...

Chapitre 1

Modélisation des interconnexions

Sommaire

Résumé

L'un des objectifs de ces travaux est d'estimer à haut niveau la consommation du réseau d'interconnexions. A ce niveau d'abstraction, un compromis temps d'estimation/précision/complexité doit être fait. Ceci nous a donc conduit à utiliser des modèles physiques d'ordre réduit pour la modélisation des lignes. Le premier chapitre de la thèse se propose d'aborder la modélisation d'un bus à l'aide de modèles physiques des différents éléments entrant dans sa composition. Le fil sous forme de modèles résistifs et capacitifs distribués a d'abord été défini, puis, au niveau bus, les buffers ainsi que les diaphonies capacitives entre fils ont été pris en compte.

1.1 Modélisation physique du fil

Nos travaux de recherche sur la modélisation de la consommation des interconnexions se situent notamment au niveau physique afin d'obtenir des modèles de consommation et de délai très fiables. Les grandeurs physiques [1] qui permettent de modéliser le comportement du fil sont au nombre de trois :

- R, la résistance du fil ;
- C, la capacité du fil ;
- I, l'inductance du fil.

Ces grandeurs dépendent des caractéristiques du fil (sa composition métallique : cuivre, aluminium...) ainsi que de ses dimensions (hauteur H, largeur W, longueur L, et donc du niveau de métal utilisé).

L'inductance (I), n'a d'impact que pour des technologies dites très submicroniques ($\leq 45nm$), pour des fils extrèmement longs [DP98] où les variations de courant sont importantes et rapides ou pour des fréquences de fonctionnement élevées. On trouve typiquement dans cette catégorie les lignes d'alimentation et la distribution de l'arbre d'horloge.

Les simulations présentées dans le mémoire satisfont les conditions proposées dans [IFN99] qui définissent des plages de variations de la longueur des interconnexions pour lesquelles les effets dus à l'inductance sont négligeables. En effet, la longueur de l'interconnexion doit alors satisfaire la double inégalité suivante pour que l'inductance soit prise en compte :

$$\frac{T_r}{2\sqrt{I_{fil/m}C_{fil/m}}} < L < \frac{2}{R_{fil/m}}\sqrt{\frac{I_{fil/m}}{C_{fil/m}}} \tag{1.1}$$

- T_r représente le temps de montée (ou de descente) de la transition à l'entrée du fil ;
- $R_{fil/m}$ représente la résistance par unité de longueur du fil ;
- $C_{fil/m}$ représente la capacité par unité de longueur du fil ;
- $I_{fil/m}$ représente l'inductance par unité de longueur du fil ;
- L représente la longueur du fil.

Dans le cas où cette inégalité ne possède pas de solution et si le temps de montée de la transition sur le fil satisfait l'inéquation suivante alors l'inductance peut être négligée :

$$T_r > 4\frac{I_{fil/m}}{C_{fil/m}} \tag{1.2}$$

Vu le contexte technologique de cette thèse, nous ne nous situons pas dans le cas où l'inductance doit être prise en compte. Les grandeurs élémentaires permettant de caractériser le fil dans les *Design Kits* des constructeurs sont au nombre de trois :

- $R_\square = \frac{\rho}{T}$, résistance par carré, exprimée en Ohm par carré [$\Omega/\square$] avec ρ la résistivité du métal et T l'épaisseur du fil ;
- C_{sq}, capacité élémentaire de la face inférieure du fil par rapport au substrat (i.e. le substrat étant définit dans la suite du document, comme étant le support de silicium sur lequel sont

[1] Les unités des grandeurs sont définies dans le tableau 4.9 en Annexe.

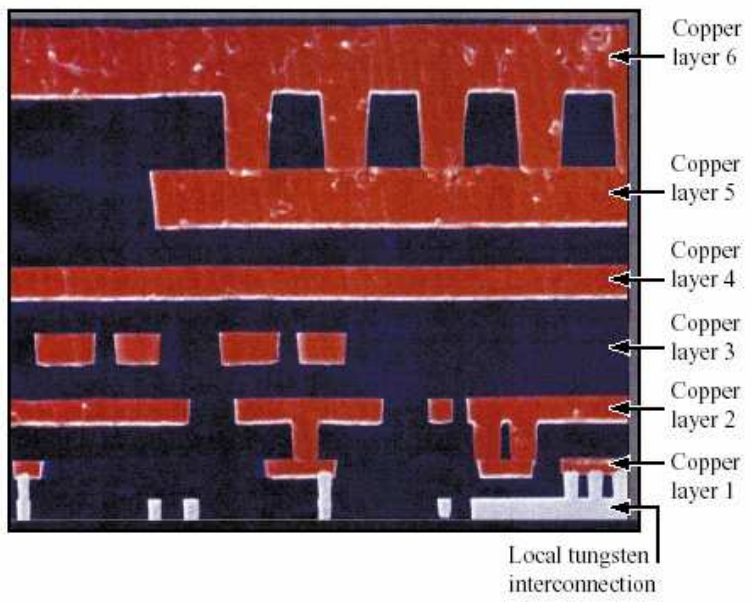


FIG. 1.1 – Réseau d'interconnexions vu en coupe [The00].

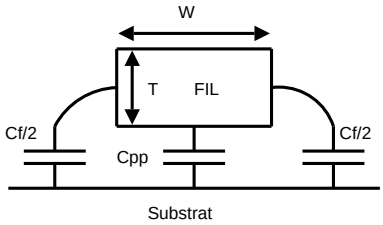


FIG. 1.2 – Détail des contributions des capacités d'un fil.

empilés les couches métalliques), exprimée en Farad par mètre $[F/m]$;

– C_e, capacité élémentaire des côtés du fil par rapport au substrat, exprimée en Farad par mètre $[F/m]$.

A l'aide de ces trois grandeurs, il est possible de calculer la résistance ainsi que la capacité du fil en fonction de ses dimensions (sa longueur L et sa largeur W exprimées en [m]). La figure 1.1 présente la vue en coupe d'un réseau d'interconnexions en fonction des différents niveaux de métal. Il apparait alors évident que la capacité de la face inférieure du fil par rapport au substrat C_{sq} ainsi que la capacité des côtés du fil par rapport au substrat C_e vont varier en fonction de la hauteur H de l'interconnexion par rapport au substrat, donc en fonction de la couche de métal utilisée. La résistance globale du fil est donnée par l'équation suivante :

$$R = R_{\square}.\frac{L}{W} \tag{1.3}$$

La capacité globale du fil C_s est la somme de deux capacités : la capacité globale de la partie inférieure du fil par rapport au substrat *(parallel-plate capacitance)* notée C_{pp} et la capacité globale des côtés du fil par rapport au substrat *(fringing capacitance)* notée C_f. Ces capacités sont représentées sur la figure 1.2. Les capacités C_{pp} et C_f sont données par les équations suivantes :

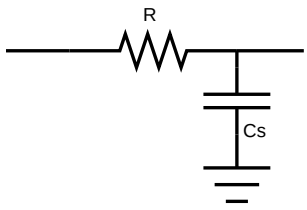


FIG. 1.3 – Modèle *lumped* ou RC d'un fil.

$$C_{pp} = C_{sq}.W.L \tag{1.4}$$

$$C_f = 2.C_e.L \tag{1.5}$$

Le facteur 2 dans l'équation de C_f est destiné à inclure le fait que les deux côtés du fil contribuent à la capacité des bords. On obtient alors la capacité globale du fil par rapport au substrat qui vaut :

$$C_s = L.[C_{sq}.W + 2.C_e] \tag{1.6}$$

Maintenant que l'on dispose des valeurs de ces grandeurs, il faut savoir quel modèle d'interconnexion va être utilisé, c'est à dire qu'il faut connaître comment seront distribuées ces grandeurs afin de modéliser le mieux possible le comportement du fil. Les modèles d'interconnexions peuvent être simples (modèle *lumped*) ou plus complexes (modèles distribués) [RCN03], ce qui va être illustré dans la suite.

1.1.1 Modèle *lumped*

Le modèle *lumped* ou modèle RC est le modèle d'interconnexion le plus simple ; il consiste à mettre bout à bout une résistance et une capacité dont les valeurs sont celles de R et C_s tels que le montre la figure 1.3. Ce modèle a l'avantage de permettre l'obtention de résultats de simulation très rapidement mais il n'est pas vraiment précis.

1.1.2 Modèles distribués

Les modèles distribués peuvent être de différentes structures, les modèles Π_n et les modèles τ_n (avec n le nombre d'étages du modèle). La figure 1.4 présente la répartition de la capacité et de la résistance en fonction de la structure choisie et du nombre d'étages.

Pour un modèle Π_n :

- les résistances valent $\frac{R}{n}$;
- les capacités des extrémités valent $\frac{C}{2.n}$;
- les capacités centrales valent $\frac{C}{n}$.

Pour un modèle τ_n :

- les résistances des extrémités valent $\frac{R}{2.n}$;
- les résistances centrales valent $\frac{R}{n}$;
- les capacités valent $\frac{C}{n}$.

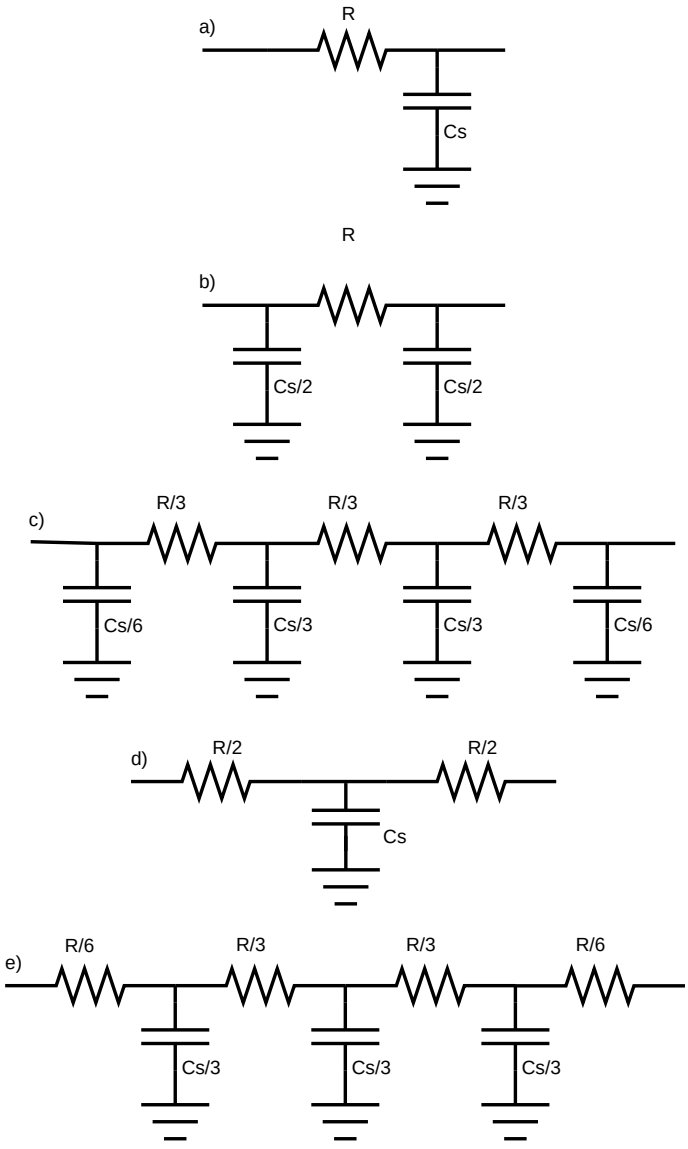


FIG. 1.4 – Modèles d'interconnexions : a)lumped b)Π c)Π_3 d)τ e)τ_3.

1.1.3 Expérimentations

Afin de déterminer le modèle à utiliser pour les expérimentations, diverses structures d'interconnexions ont été testées avec un simulateur SPICE tel que le montre la figure 1.5. Le graphique a) de cette figure montre que pour de courtes interconnexions (1mm ici), la réponse à une rampe (une transition à l'entrée du fil) est identique quel que soit le modèle physique d'interconnexion utilisé. Le graphique b) de cette figure représente la réponse à une rampe pour une interconnexion plus longue (10mm ici) ; il est important de remarquer que le modèle lumped n'est plus utilisable car totalement imprécis (environ 35% de différence avec un modèle Π_3 par exemple).
Afin de déterminer jusqu'à quel nombre d'étages le modèle physique peut être fractionné, regardons la différence entre un modèle Π à 3 étages et à 10 étages (graphique c)). Une infime différence (moins de 1% sur un fil de 10mm) justifie le fait qu'il n'est pas nécessaire de fractionner l'interconnexion sur un grand nombre d'étages afin d'obtenir des résultats précis. De plus, il est important de souligner que plus le nombre d'étages est grand et plus les simulations sont longues. D'après [RCN03], les valeurs obtenues pour un modèle Π_3 sont éloignées de moins de 3 % des valeurs expérimentales, ce qui se confirme par nos expérimentations.
Puisque le modèle Π_3 est relativement simple et précis tant pour des interconnexions courtes que longues, c'est ce modèle que nous avons retenu pour les modèles futurs.
La section suivante va maintenant s'attacher à modéliser le bus dans sa totalité.

1.2 Modélisation physique du bus

Un bus n bits est constitué de n fils parallèles de même longueur, ainsi que, au minimum de $2n$ buffers (n buffers à l'entrée du bus et n à la sortie). Si des techniques de bufferisation du bus sont utilisées afin d'augmenter les performances temporelles, d'autres buffers seront alors nécessaires.

1.2.1 Modélisation physique des buffers

Les buffers (ou inverseurs : un inverseur est composé de deux transistors *MOS*) ont pour rôle de regénérer le signal qui peut s'atténuer lors de la transmission sur la ligne. Les bus peuvent être bufferisés de différentes manières.
Pour des lignes courtes où le signal se propage rapidement, on utilise en général deux buffers ; un au départ de chaque ligne du bus (*driver buffer*) et un à l'arrivée (*receiver buffer*).
Pour des lignes longues où le signal s'atténue et où le temps de propagation devient critique, il est nécessaire de regénérer le signal afin d'accélérer sa propagation en utilisant des méthodes d'insertion de buffer (méthodes qui seront expliquées section 1.3.3). Afin de modéliser ces techniques d'insertion de buffer, il est nécessaire de connaître leurs paramètres résistifs et capacitifs tels que définis dans [RCN03].

Résistance

La résistance d'un transistor *MOS* est déterminée par l'importance de la taille du transistor ; c'est une somme de plusieurs résistances parasites qui sont (figure 1.6) :

– la résistance du canal R_{canal} ;

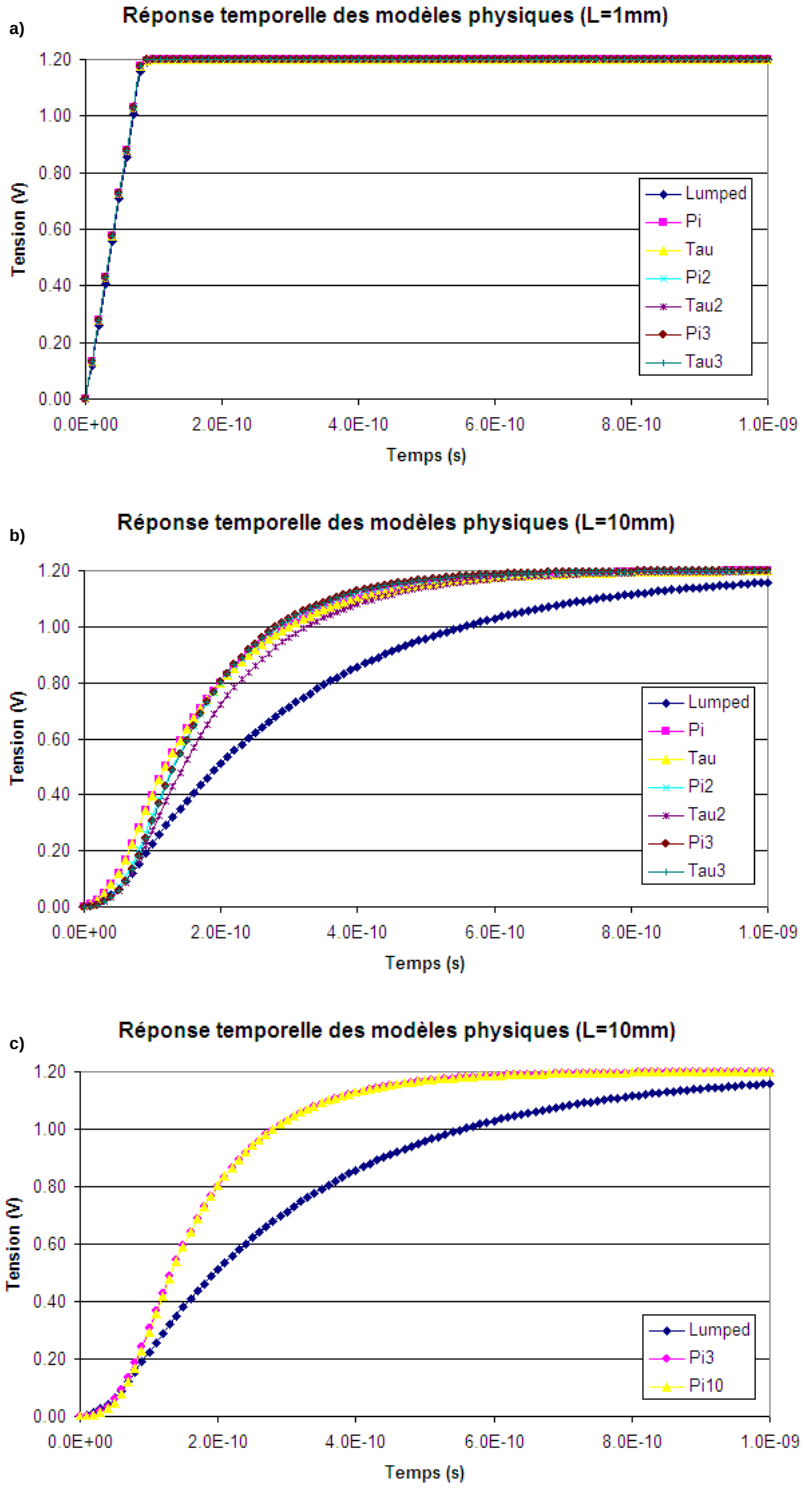


FIG. 1.5 – Réponses temporelles des différents modèles d'interconnexions.

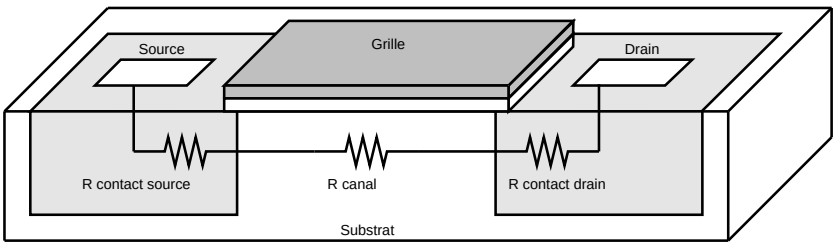


FIG. 1.6 – Résistances parasites d'un transistor *MOS*.

- la résistance de contact de la source $R_{contactsource}$;
- la résistance de contact du drain $R_{contactdrain}$.

Les résistances de contact de source et de drain sont généralement considérées comme étant les mêmes puisque le transitor *MOS* est symétrique.

$$R_{MOS} = R_{canal} + R_{contactsource} + R_{contactdrain} = R_{canal} + 2R_{contactsource} \tag{1.7}$$

La résistance du canal peut être obtenue grâce à la formule suivante :

$$R_{canal} = \frac{\frac{L_{MOS}}{W_{MOS}}}{\mu C_{ox}(V_{dd} - V_t)} \tag{1.8}$$

- L_{MOS} représente la longueur du canal (i.e. la valeur de la technologie employée) ;
- W_{MOS} représente la largeur du transitor ;
- μ représente la mobilité des porteurs (i.e. les électrons pour un *NMOS*, les trous pour un *PMOS*) ;
- C_{ox} représente la capacité d'oxyde par unité de surface ;
- V_{dd} représente la tension d'alimentation ;
- V_t représente la tension de seuil du transistor.

Les résitances de contact sont obtenues grâce à la formule suivante :

$$R_{contactsource} = R_{contactdrain} = \frac{\rho_{contact}}{S_{contact}} \tag{1.9}$$

- $\rho_{contact}$ représente la résistivité du contact ;
- $S_{contact}$ représente la surface du drain ou de la source.

Un inverseur est composé d'un transistor *NMOS* et d'un transitor *PMOS* ; la résistance du transistor *NMOS* est active lorsque l'on effectue une transition montante et celle du *PMOS* pour une transition descendante. Or la mobilité des trous μ_p est environ deux fois inférieure à celle des électrons μ_n. Donc afin d'équilibrer les temps de descente et de montée de l'inverseur, le transitor *PMOS* est généralement deux fois plus gros que le transitor *NMOS* (i.e. $W_{PMOS} \approx 2W_{NMOS}$).
La résistance totale de l'inverseur est donc approximée par la formule suivante :

$$R_{buffer} = \frac{1}{2}\left(R_{NMOS} + R_{PMOS}\right) \tag{1.10}$$

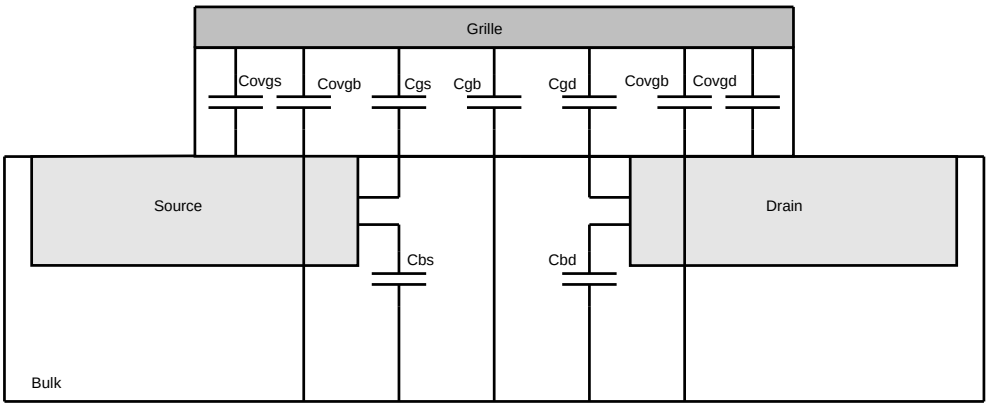


FIG. 1.7 – Capacités parasites d'un transistor *MOS*.

Capacités

Il est important de connaître toutes les capacités parasites qui entrent dans la composition de l'inverseur. En effet, ces capacités vont conditionner les performances (temporelles et en consommation) des lignes d'interconnexion et vont également servir de paramètres aux techniques d'insertion de buffers (amélioration des performances temporelles).
La figure 1.7 présente la composition des capacités parasites d'un transistor *MOS*.

Capacités parasites des jonctions Les capacités parasites de jonction C_{BS} et C_{BD} sont les capacités qui interviennent dans les régions de déplétion entre la source et le substrat (*Bulk*) et le drain et le substrat.
Elles sont constituées de deux capacités ; l'une qui est la capacité de la face inférieure du plot de drain (ou de source) $C_{bottom-plane}$ par rapport au substrat et l'autre qui est la capacité des côtés du plot de drain (ou de source) $C_{side-wall}$ par rapport au substrat et dont les équations sont les suivantes :

$$C_{bottom-plane} = C_J A_D \left[1 - \frac{V_{BS}}{PB}\right]^{-MJ} \tag{1.11}$$

- C_J représente la capacité de jonction sous polarisation nulle par unité de surface ;
- A_D représente la surface du plot de drain (ou de source) ;
- V_{BS} représente la tension à travers la jonction ;
- PB représente le potentiel de contact de la jonction ;
- MJ représente le coefficient de gradient pour la jonction sous le plot.

$$C_{side-wall} = C_{JSW} P_D \left[1 - \frac{V_{BS}}{PB}\right]^{-MJSW} \tag{1.12}$$

- C_{JSW} représente la capacité de jonction des côtés sous polarisation nulle par unité de surface ;
- P_D représente le périmètre du plot de drain (ou de source) ;
- V_{BS} représente la tension à travers la jonction ;

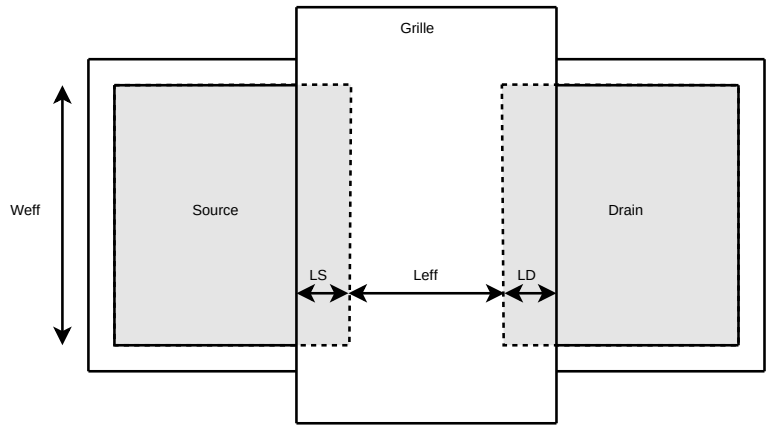


FIG. 1.8 – Représentation des distances de recouvrement.

– PB représente le potentiel de contact de la jonction ;
– $MJSW$ représente le coefficient de gradient pour la jonction sur les côtés du plot.

Les capacités de jonction C_{BS} et C_{BD} peuvent donc se définir comme la somme des deux précédentes capacités :

$$C_{BD} = C_{BS} = C_{side-wall} + C_{bottom-plane} \tag{1.13}$$

Capacités parasites de la grille Les capacités parasites de la grille se décomposent en deux catégories : les capacités fixes qui sont des capacités parasites dues au recouvrement (*overlap capacitance*) de la grille sur les zones de drain et de source :

– C_{OVGB} la capacité de recouvrement entre la grille et le substrat ;
– C_{OVGD} la capacité de recouvrement entre la grille et le drain ;
– C_{OVGS} la capacité de recouvrement entre la grille et la source.

et les capacités qui varient selon la région de fonctionnement du transistor :

– C_{GB} la capacité entre la grille et le substrat ;
– C_{GD} la capacité entre la grille et le drain ;
– C_{GS} la capacité entre la grille et la source.

Les capacités de recouvrement sont fonction des longueurs et largeurs effectives du canal et des plots de drain (ou de source) tel que le montre la figure 1.8.

Le calcul des capacités de recouvrement s'effectue alors de la manière suivante :

$$C_{OVGB} = L_{eff} C_{GB0} \tag{1.14}$$

$$C_{OVGD} = W_{eff} C_{GD0} \tag{1.15}$$

$$C_{OVGS} = L_{eff} C_{GS0} \tag{1.16}$$

– L_{eff} représente la longueur effective du canal ;
– W_{eff} représente la largeur effective du canal ;

Zone de fonctionnement	C_{GB}	C_{GD}	C_{GS}
Bloqué	$C_{ox}W_{eff}L_{eff}$	0	0
Linéaire	0	$\frac{1}{2}C_{ox}W_{eff}L_{eff}$	$\frac{1}{2}C_{ox}W_{eff}L_{eff}$
Saturé	0	$\frac{2}{3}C_{ox}W_{eff}L_{eff}$	0

TAB. 1.1 – Capacités parasites de grille en fonction de la zone de fonctionnement du transistor

- L_D représente la longueur de recouvrement sur le plot de drain ;
- L_S représente la longueur de recouvrement sur le plot de source ($L_D = L_S$) ;
- C_{GB0} représente la capacité grille/substrat sous polarisation nulle par unité de longueur ;
- C_{GD0} représente la capacité grille/drain sous polarisation nulle par unité de longueur ;
- C_{GS0} représente la capacité grille/source sous polarisation nulle par unité de longueur.

Les capacités de grille (C_{GB}, C_{GD} et C_{GS}), quant à elles, dépendent de la zone de fonctionnement du transistor et sont données dans le tableau 1.1. Dans ce tableau, C_{ox} représente la capacité d'oxyde vue sous la grille du transistor, calculée par la formule :

$$C_{ox} = \frac{\varepsilon_{ox}\varepsilon_0}{T_{ox}} \tag{1.17}$$

- ε_{ox} représente la permittivité du diélectrique (oxyde) ;
- ε_0 représente la permittivité du vide ;
- T_{ox} représente l'épaisseur d'oxyde sous la grille.

La capacité totale de la grille d'un transistor peut donc être calculée de la manière suivante :

$$C_G = (C_{GB} ou C_{GD} ou C_{GS}) + 2C_{OVGB} + C_{OVGD} + C_{OVGS} \tag{1.18}$$

$$C_G \approx C_{ox}W_{eff}L_{eff} + 2C_{OVGB} + C_{OVGD} + C_{OVGS} \tag{1.19}$$

Capacités de l'inverseur Maintenant que nous avons étudié les capacités parasites qui entrent en jeu dans un transistor *MOS*, nous allons présenter comment déterminer les capacités d'entrée (C_{IN}) et de sortie (C_{OUT}) de l'inverseur (figure 1.9). Ces capacités sont fonction de la taille des transistors *NMOS* et *PMOS* qui composent l'inverseur.
Pour les mêmes raisons que précédemment (respect des temps de montée et de descente égaux), la largeur du transistor *PMOS* sera deux fois supérieure à celle du transitor *NMOS*. Nous supposerons donc que la capacité parasite de grille du transistor *PMOS* est deux fois plus importante que celle du transitor *NMOS*.

$$C_{G_{PMOS}} = 2C_{G_{NMOS}} \tag{1.20}$$

Il en va de même pour les capacités parasites de jonction :

$$C_{BD_{PMOS}} = 2C_{BD_{NMOS}} \tag{1.21}$$

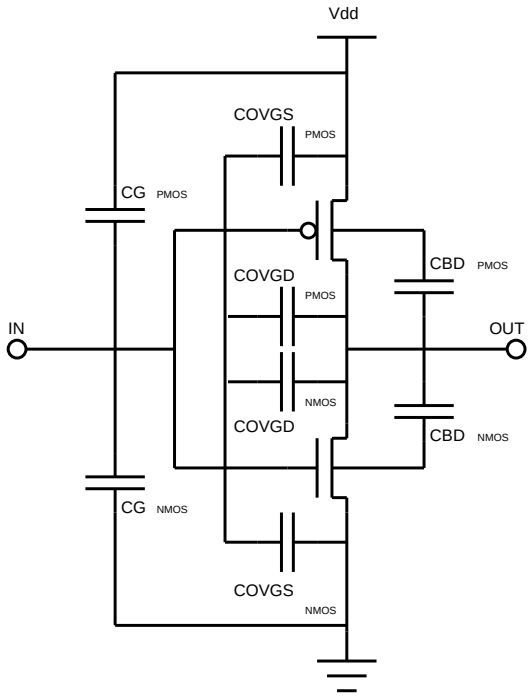


FIG. 1.9 – Capacités parasites de l'inverseur.

Ce qui permet d'écrire les équations des capacités d'entrée et de sortie de l'inverseur :

$$C_{IN} = C_{G_{PMOS}} + C_{G_{NMOS}} = 3C_{G_{NMOS}} \tag{1.22}$$

$$C_{OUT} = C_{BD_{PMOS}} + C_{BD_{NMOS}} = 3C_{BD_{NMOS}} \tag{1.23}$$

Toutes les valeurs des paramètres technologiques qui viennent d'être définis dans cette partie et qui permettent de modéliser le comportement des transistors, sont présentes dans les librairies des *DesignKit* des transistors *MOS* de chaque technologie (voir Annexes 4.5).

1.2.2 Regroupement des lignes : le *crosstalk*

Généralement, les lignes d'interconnexions sont regroupées afin d'obtenir un bus de données permettant la transmission d'informations entre blocs. Le fait de regrouper les lignes d'interconnexions en un bus peut faire apparaitre un nouveau phénomène de diaphonie capacitive que l'on nomme le *crosstalk*. Ce phénomène apparaît si les lignes ne sont pas excessivement éloignées les unes des autres.
La capacité de *crosstalk* entre deux fils adjacents dépend quant à elle de la surface en regard entre ces deux fils, et donc de l'épaisseur T du fil, de sa longueur L ainsi que de l'espacement S entre ceux ci :

$$C_c = \varepsilon_{ox} \frac{TL}{S} \tag{1.24}$$

– ε_{ox} représente la permittivité du diélectrique (oxyde) ;

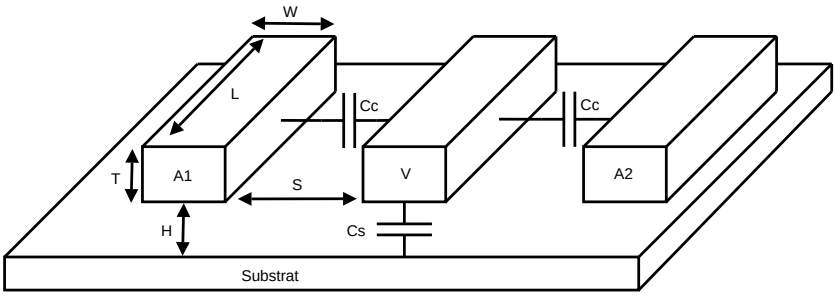


FIG. 1.10 – Un fil victime V soumis au couplage capacitif de ses deux agresseurs *A1* et *A2*.

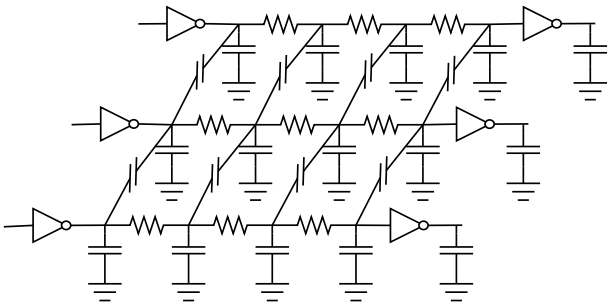


FIG. 1.11 – Modèle complet Π_3 pour 3 fils avec couplage *crosstalk*.

- T représente l'épaisseur du fil (*Thickness*) ;
- L représente la longueur du fil (*Length*) ;
- S représente l'espacement entre deux fils (*Spacing*).

Lors de transition sur des fils adjacents, il y a génération d'un bruit parasite sur le fil victime (ie : le fil central par rapport à ses voisins de gauche et de droite) dû au couplage entre les fils.
Le bruit dû à la diaphonie capacitive est relativement localisé. On modélise en général un système soumis au *crosstalk* en réduisant les ordres supérieurs au premier : ainsi on ne considère que trois fils comme le montre la figure 1.10

Le couplage capacitif entre les fils peut également se répartir sur les noeuds du modèle distribué Π_3 qui a été présenté précédemment (section 1.1.2) ; le modèle complet du bus qui est alors obtenu est représenté sur la figure 1.11. Il est également possible qu'une diaphonie capacitive soit introduite avec les lignes de la couche métallique supérieure et/ou inférieure. Généralement les lignes d'interconnexions sont routées orthogonalement d'une couche métallique à l'autre tel qu'illustré à la figure 1.12. Les simulations que nous avons réalisées (suivant les dimensions technologiques des interconnexions de nos *Design Kits*) montrent qu'il existe un rapport d'environ 10^4 entre la capacité du fil par rapport au substrat (C_s) et les capacitées de croisement entre les couches (notées C_{1-2} sur la figure 1.12). Donc, l'importance de la diaphonie capacitive inter-couches reste très faible par rapport à celle intra-couche où les lignes sont parallèles sur une grande distance. La partie suivante présentera les problèmes liés à l'évolution des paramètres résistifs et capacitifs ainsi que les problèmes qui en découlent, notamment ceux liés au *crosstalk*.

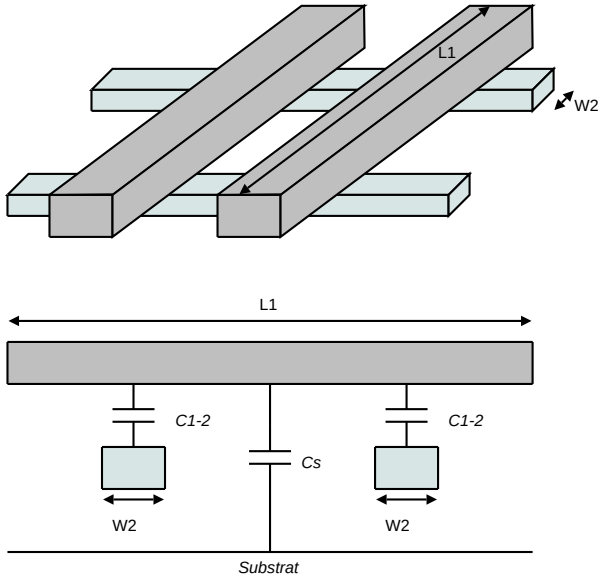


FIG. 1.12 – Diaphonie capacitive entre les différentes couches de métal.

Année	2001	2003	2005	2007	2009	2011	2013	2015
Technologie (nm)	130	100	80	65	50	40	32	25
Densité (Mtransitor/cm^2)	39	61	97	154	245	389	617	980
Frequence (MHz)	1684	2976	5204	9285	12369	17658	22980	33403
Tension d'alimentation (V)	1.2-1.1	1.2-1.0	1.1-0.9	1.1-0.8	1.0-0.8	1.0-0.7	0.9-0.6	0.9-0.6
Couches de métal	10	13	15	15	16	16	17	17
[1]Délai (ps)	x	191	307	486	783	1224	1572	5951
Puissance (W)	130	149	167	189	210	225	251	270
[2]Longueur routage (m/cm^2)	x	579	1019	1439	2000	2500	3125	4000
Constante diélectrique	3.0-3.6	3.3-3.6	3.1-3.4	2.7-3.0	2.5-2.8	2.5-2.8	2.1-2.4	1.9-2.2

TAB. 1.2 – Evolution des paramètres technologiques tirés de [ITR02][ITR04] et [ITR06] ([1] Le délai est calculé pour un fil de longueur typique de 1mm sur la couche de métal 1) ([2] La longueur de routage est calculée pour 6 couches de métal sur la totalité des couches de métal présentes dans la technologie)

1.3 Evolution des paramètres technologiques et conséquences

Cette partie présente comment se traduit l'évolution des paramètres technologiques (notamment les résistances et les capacités) et quelles en sont les conséquences sur les performances des interconnexions en termes de délai et de consommation.

Les prévisions de l'ITRS [ITR02, ITR04, ITR06](voir tableau 1.2 ainsi que figure 1.13) montrent une diminution de la finesse de gravure ainsi qu'une augmentation du nombre de transitors par puce. Ces évolutions corrélées avec une augmentation de la fréquence de fonctionnement des circuits entraînent une forte augmentation de la puissance consommée. Afin d'essayer de minimiser cette augmentation, un effort est porté sur la diminution de la tension d'alimentation (figure 1.13).

Avec la diminution de la technologie, les procédés de fabrication deviennent de plus en plus complexes et il devient de moins en moins facile de maîtriser les critères de délai et de consommation. L'évolution des dimensions des transistors et des fils se traduit par une évolution du comportement du circuit, tout particulièrement au niveau temporel. Ainsi le délai d'un fil devient supérieur

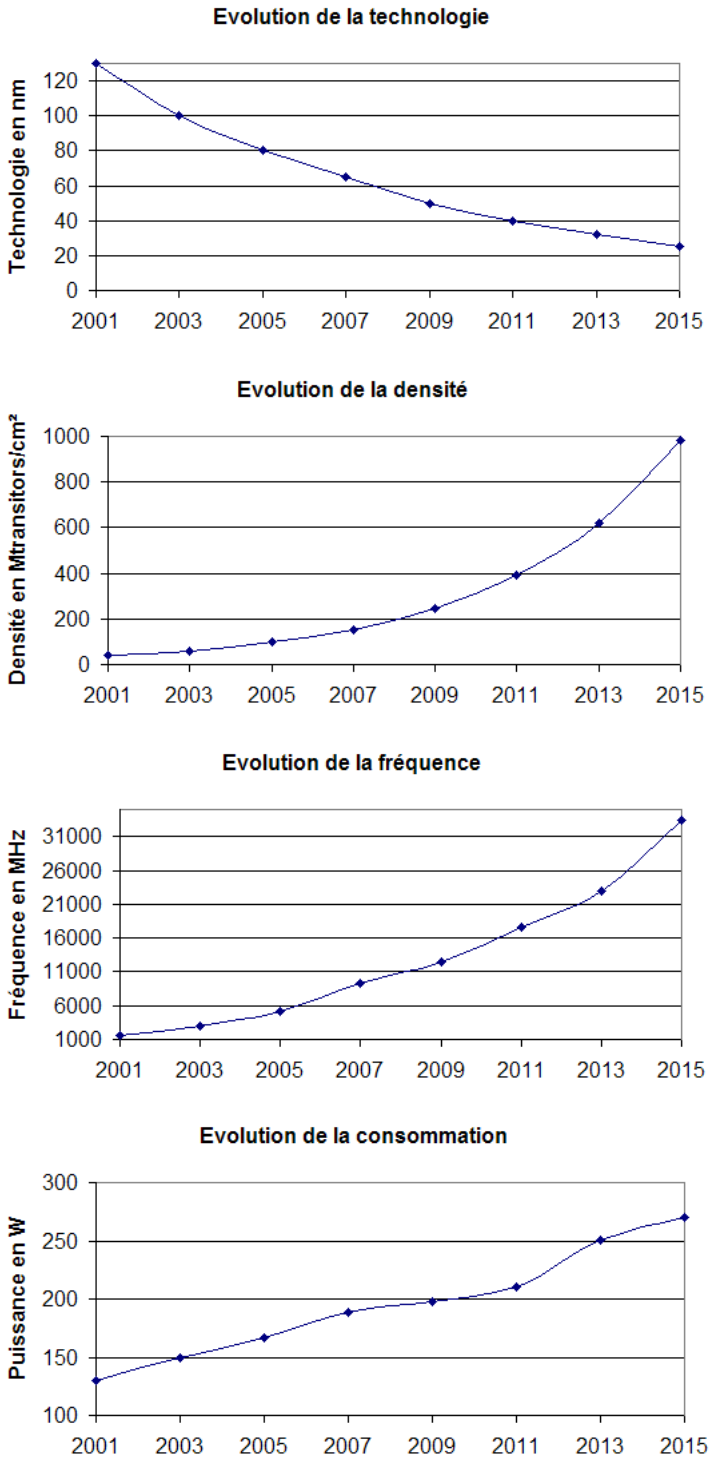


FIG. 1.13 – Evolution des grandeurs technologie, densité, fréquence et consommation ([ITR02][ITR04] et [ITR06]).

à celui d'une porte [HMH01].
Nous observons de plus, une forte augmentation de la densité du réseau d'interconnexion. Afin de contenir l'explosion du nombre d'interconnexions, nous observons parallèlement une augmentation du nombre de couches de métal. Les procédés de fabrication, notemment ceux des interconnexions, requierent une planarisation de la surface par une phase de décapage mécano-chimique avant l'apport de nouvelle matière. En ce qui concerne un niveau de metal, il peut être nécessaire de l'homogénéiser, c'est à dire de combler les endroits où il n'y a rien par l'insertion d'interconnexions appelées *dummies*. Ce type d'interconnexions n'a pas vocation à transporter de l'information mais sert simplement à homogénéiser/rigidifier l'ensemble pour que la phase de décapage mécano-chimique n'attaque pas les couches inférieures. Les *dummies* peuvent alors être source de diaphonie capacitive avec les interconnexions classiques tel qu'illustré dans [BFB+04],[KKI+05]. Cette diaphonie, qui apparaît avec les lignes du bus qui n'ont qu'un seul voisin (i.e. le bit de poids fort et le bit de poids faible) est prise en compte lors de l'estimation de consommation. En effet, la consommation du bit de poids fort/faible est calculée en positionnant un agresseur virtuel à un niveau logique stable (voir Chapitre suivant).
Dans les technologies plus anciennes (où la largeur de grille n'était pas inférieure à 250 nm), les interconnexions étaient réalisées en aluminium et séparées les unes des autres par un isolant (SiO2) dont la constante diélectrique avoisinait 4. Avec l'évolution des procédés de fabrication, l'aluminium s'est vu remplacé par le cuivre dont la conductivité est bien meilleure. Grâce à cette évolution, le délai de propagation sur une interconnexion a fortement diminué. En parallèle à ce remplacement de l'aluminium par du cuivre, des isolants à faible permittivité (*LowK*) ont fait leur apparition pour se substituer au SiO2. Ces évolutions ont permis d'augmenter de façon significative les performances temporelles des interconnexions. L'atout principal de l'utilisation d'isolant à faible permittivité est de permettre de réduire les phénomènes de diaphonie capacitive entre les fils (*crosstalk*) ainsi qu'entre les fils et le substrat.
Les deux sous parties suivantes présenteront comment se traduit l'évolution des paramètres résistifs et capacitifs sur le délai et la consommation des interconnexions.

1.3.1 Evolution des paramètres résistifs

Comme le montre la figure 1.14, le délai sur un fil de longueur constante entre différentes technologies ne cesse d'augmenter ; ce phénomène est dû entre autre à l'évolution de la résistance de ces fils. En effet entre deux technologies, les dimensions des fils diminuent et la résistance qui est inversement proportionnelle à la section du fil se voit augmentée.

1.3.2 Evolution des paramètres capacitifs

Les capacités parasites ne cessent d'augmenter ; en effet, on peut noter que les dimensions physiques des fils ont changé. Les fils deviennent plus hauts que larges tel que le montre le tableau 1.3, ce qui favorise l'augmentation de la capacité de *crosstalk* entre les fils. Afin de compenser cette augmentation, des diélectriques à faible permittivité sont utilisés dans le but de réduire la capacité de couplage. La suite de cette partie expliquera plus précisement les problèmes liés au couplage capacitif.

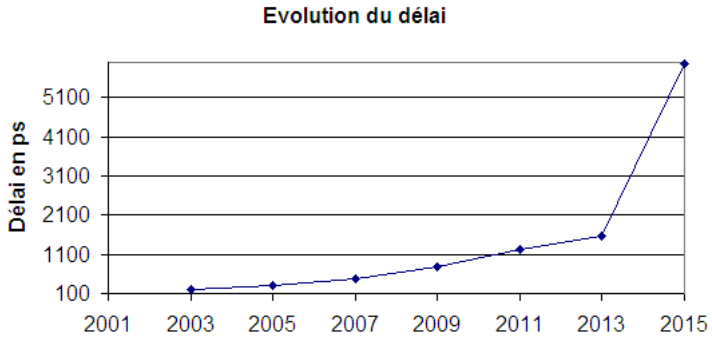


FIG. 1.14 – Evolution du délai (Métal1, longueur 1mm).

Technologie (nm)	Largeur (μm)	Epaisseur (μm)	Rapport d'aspect (L/E)
180	0.8	1.25	1.56
130	0.6	1.2	2
90	0.5	1.2	2.4
65	0.45	1.2	2.66

TAB. 1.3 – Evolution des dimensions des fils et du rapport d'aspect entre les couches de métal [PTM07]

Le *crosstalk* : source d'erreurs

Comme nous l'avons vu dans la section sur la modélisation physique du bus, il existe un couplage capacitif entre les fils. Lorsqu'une ligne commute, la ligne voisine est perturbée ; on définit alors la ligne perturbée comme la victime et la ligne perturbatrice comme l'agresseur. Il existe deux catégories de couplage :

- le couplage positif, lorsque l'amplitude du bruit dépasse d'une valeur positive la tension sur le fil victime ;
- le couplage négatif, lorsque l'amplitude du bruit dépasse d'une valeur négative la tension sur le fil victime.

Sur la figure 1.15, les différents types de couplage positifs et négatifs sont représentés.

- Sur la figure 1.15 a), l'agresseur effectue une transition montante alors que la victime reste à *GND*. Le cas sans couplage représenté en pointillés montre que la victime n'est pas perturbée. Avec couplage on observe un bruit positif au dessus de *GND*.
- Sur la figure 1.15 b), l'agresseur effectue une transition montante alors que la victime reste à V_{dd}. Avec couplage on observe un bruit positif au dessus de V_{dd}.
- Sur la figure 1.15 c), l'agresseur effectue une transition descendante alors que la victime reste à *GND*. Avec couplage on observe un bruit négatif en dessous de *GND*.
- Sur la figure 1.15 d), l'agresseur effectue une transition descendante alors que la victime reste à V_{dd}. Avec couplage on observe un bruit négatif en dessous de V_{dd}.

Ce sont les cas a) et d) qui sont les plus néfastes. En effet, le pic de tension au dessus de *GND* (pour le cas a)) et en dessous de V_{dd} (pour le cas d)) peut être source d'erreur dans le cas où ce pic parvient jusqu'à la tension de seuil du buffer en sortie du bus.

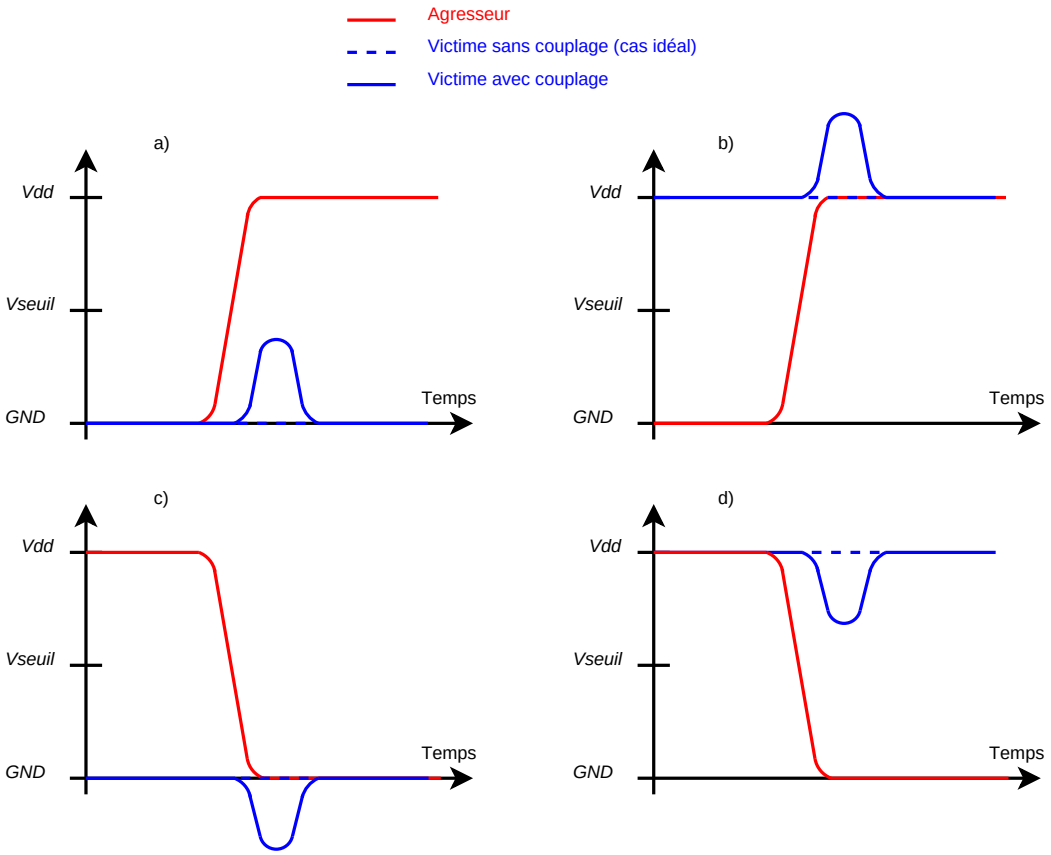


FIG. 1.15 – Bruits sur le fil victime (sans transition) en fonction du type de couplage.

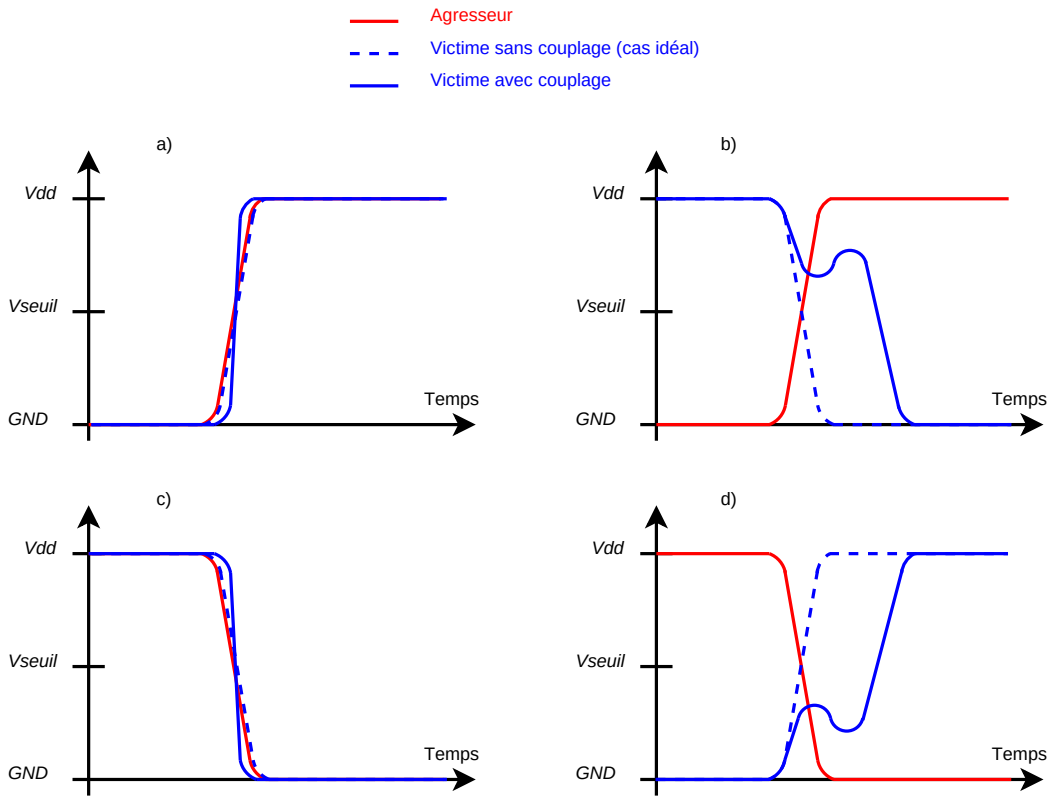


FIG. 1.16 – Bruits sur le fil victime (avec transition) en fonction du type de couplage.

Le bruit généré sur la ligne victime n'est pas le seul phénomène introduit par le *crosstalk*. Nous avons présenté le cas où le fil victime reste à un niveau logique stable pendant que son ou ses agresseurs commutent. Nous allons maintenant expliquer ce qui se passe lorsque les fils agresseurs et le fil victime effectuent des transitions simultanées.

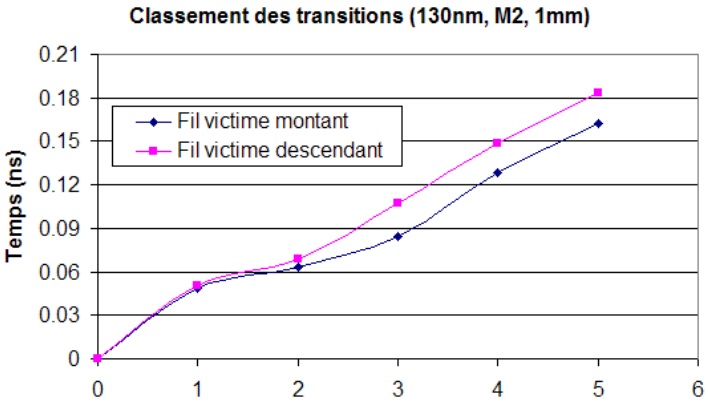
Le *crosstalk* : source de retard

La partie précédente a montré qu'une transition sur un fil affecte son voisin en créant un pic de tension sur celui-ci. Lors de transitions simultanées sur le fil victime et agresseur, un pic de tension sera également généré ; ce pic peut selon la configuration des transitions légèrement accélérer ou considérablement ralentir la propagation de la donnée sur le fil victime (figure 1.16). Il existe quatre cas de couplage possibles.

- Sur la figure 1.16 a), l'agresseur et la victime effectuent des transitions montantes. Le cas sans couplage représenté en pointillés montre que les transitions se font identiquement. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens de la transition ; les transitions sur le fil victime et agresseur vont en quelque sorte s'accélérer mutuellement.
- Sur la figure 1.16 b), l'agresseur effectue une transition montante alors que la victime effectue une transition descendante. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens inverse de la transition du fil ; la transition sur le fil victime va donc être ralentie.

- Sur la figure 1.16 c), l'agresseur et la victime effectuent des transitions descendantes. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens de la transition ; les transitions sur le fil victime et agresseur vont en quelque sorte s'accélérer mutuellement.
- Sur la figure 1.16 d), l'agresseur effectue une transition descendante alors que la victime effectue une transition montante. Avec couplage, la transition sur le fil agresseur va générer sur le fil victime un pic de tension dans le sens inverse de la transition du fil ; la transition sur le fil victime va donc être ralentie.

Lors des transitions, le fil victime va être parasité par les agresseurs ; ceux ci peuvent être au nombre de un ou deux. En effet, pour un fil situé au milieu du bus, ses voisins sont au nombre de deux, en revanche un fil situé aux extrémités du bus n'a qu'un seul voisin.
Il est donc possible ici d'effectuer un classement des types de transition selon la rapidité de propagation des transitions.
Ici $\uparrow$ représente une transition montante, $\downarrow$ représente une transition descendante et $-$ signifie qu'il n'y a pas de transition sur le fil.
Prenons le cas de trois fils qui commutent dans la même direction ($\uparrow\uparrow\uparrow$ ou $\downarrow\downarrow\downarrow$). La transition sur le fil victime (ie : le fil central) va se voir accélérée par les perturbations de ses deux agresseurs. Maintenant, si un de ses agresseurs ne transite pas ($-\uparrow\uparrow$ ou $-\downarrow\downarrow$), la transition sur le fil victime va se voir accélérée par la perturbation d'un seul de ses agresseurs et donc sera moins rapide que dans le cas précédent.
Prenons maintenant le cas de trois fils où le fil victime et un de ses agresseurs commutent dans la même direction tandis que son autre agresseur commute dans le sens inverse ($\uparrow\uparrow\downarrow$ ou $\downarrow\downarrow\uparrow$). Un de ses agresseurs va accélérer la transition et l'autre la ralentir. Maintenant, si l'agresseur qui commute dans le même sens que la victime, ne transite pas ($-\uparrow\downarrow$ ou $-\downarrow\uparrow$), la transition sur le fil victime va se voir ralentir par la perturbation d'un seul de ses agresseurs et donc la transition sera moins rapide que dans le cas précédent.
Poursuivons ce raisonnement en prenant le cas où les agresseurs commutent dans le sens inverse de la victime ($\downarrow\uparrow\downarrow$ ou $\uparrow\downarrow\uparrow$). Ici la victime va se voir perturbée par ses deux agresseurs et la transition sera encore plus lente que dans le cas précédent ; ceci représente le pire cas de temps de propagation sur le bus.
La figure 1.17 illustre l'augmentation du temps de propagation sur le fil victime en fonction des transitions sur ses agresseurs sur une couche de métal réservée au bus pour une longueur typique de 1mm dans une technologie 130nm. (Les résultats expérimentaux sont les mêmes pour les autres technologies qui ont été expérimentées, à savoir 90nm et 65nm).
Nous pouvons de cette manière effectuer un classement des transitions selon le temps de propagation sur le fil victime tel que le montre le tableau 1.4 où g représente le facteur de délai et r le ratio de la capacité de *crosstalk* par rapport à la capacité du fil par rapport au substrat.
Afin de vérifier la tendance de l'augmentation du délai dû aux interconnexions, nous avons représenté dans le tableau 1.5 l'évolution du paramètre r sur les différentes couches de métal dans trois technologies (130, 90 et 65nm).
Nous pouvons remarquer que le paramètre r a tendance à augmenter lors d'un changement de technologie, ceci étant dû à l'évolution du rapport d'aspect (voir tableau 1.3). Outre les pro-



Abscisse	Montant	Descendant
0	- - -	- - -
1	↑↑↑	↓↓↓
2	− ↑↑	− ↓↓
3	↓↑↑	↑↓↓
4	− ↑↓	− ↓↑
5	↓↑↓	↑↓↑

FIG. 1.17 – Variation du temps de propagation sur le fil victime en fonction du type de transition.

blèmes de bruit et de temps de propagation, le phénomène de *crosstalk* est également à l'origine de l'augmentation de la consommation, ce qui va être expliqué dans la partie suivante.

Le *crosstalk* : source de consommation

La consommation statique sur les bus peut être négligée puisque sa contribution dans la part de la consommation totale est très faible. En effet les données qui circulent sur le bus changent souvent d'état ; c'est ce nombre important de transitions (donc une activité importante) qui va faire que la consommation sur les bus est exclusivement de la consommation dynamique. De plus, le nombre de transitors présents sur les bus est faible comparé à un bloc de calcul qui effectue des opérations plus complexes ; on aura donc peu de transistors qui vont fuir.
Le calcul de la consommation dynamique se représente de la manière suivante :

$$P_{dynamique} = \sum_{i \in Nbit} \alpha_i . C_{L_i} . V_{dd} . V_{swing} . F \tag{1.25}$$

- N_{bit} représente le nombre de bits du bus considéré ;
- α_i représente l'activité du fil i ;
- C_{L_i} représente la capacité du fil i (capacité qui va être détaillée ci-dessous) ;
- V_{dd} représente la tension d'alimentation ;
- V_{swing} représente la tension d'excursion ;

C_L	Types de transition				g
C_s	$(\uparrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\downarrow)$			1
$C_s + C_c$	$(-,\uparrow,\uparrow)$	$(-,\downarrow,\downarrow)$	$(\uparrow,\uparrow,-)$	$(\downarrow,\downarrow,-)$	1+r
$C_s + 2.C_c$	$(-,\uparrow,-)$ $(\uparrow,\uparrow,\downarrow)$	$(-,\downarrow,-)$ $(\uparrow,\downarrow,\downarrow)$	$(\downarrow,\uparrow,\uparrow)$	$(\downarrow,\downarrow,\uparrow)$	1+2.r
$C_s + 3.C_c$	$(-,\uparrow,\downarrow)$	$(-,\downarrow,\uparrow)$	$(\uparrow,\downarrow,-)$	$(\downarrow,\uparrow,-)$	1+3.r
$C_s + 4.C_c$	$(\uparrow,\downarrow,\uparrow)$	$(\downarrow,\uparrow,\downarrow)$			1+4.r

TAB. 1.4 – Capacité parasite (C_L) et facteur de délai (g) du fil victime en fonction du type de transition.

	r_{130nm}	r_{90nm}	r_{65nm}
M1	0.92	0.8	0.8
M2	1.84	1.84	1.79
M3	3.06	3.37	3.42
M4	4.29	4.91	5.06
M5	2.70	6.44	6.70
M6	3.8	3.23	8.33
M7		4.47	3.13
M8			4.44

TAB. 1.5 – Evolution du paramètre r lors d'un changement de technologie pour toutes les couches de métal.

– F représente la fréquence des transitions.

Dans l'équation de la consommation dynamique, le paramètre C_L se décompose en la somme de quatre capacités :

– C_{OUT} qui représente la capacité de sortie de l'inverseur à l'entrée du fil ;
– C_s qui représente la capacité du fil par rapport au substrat ;
– C_c qui représente la capacité de *crosstalk* ;
– C_{IN} qui représente la capacité d'entrée de l'inverseur à la fin du fil.

Donc, plus la capacité de *crosstalk* C_c sera importante, plus la consommation augmentera.
Nous venons de voir que l'évolution des paramètres résistifs et capacitifs introduit une contrainte forte sur le temps de propagation des données au sein des interconnexions. C'est pourquoi lorsque ce temps devient trop critique, notamment pour des lignes d'interconnexions longues, les concepteurs de circuit ont recours à des méthodes d'insertion de buffers afin d'accélérer la propagation des données.

1.3.3 Evolution du délai : insertion de répéteurs

Cette partie va expliquer comment exprimer le temps de propagation au sein des interconnexions, puis les méthodes d'insertion de répéteurs seront présentées.

Expression du temps de propagation

Il existe deux méthodes de mesure du temps de propagation sur les interconnexions tel que le montre la figure 1.18.

– La première consiste à calculer la différence de temps entre le passage à 50% de la transition par rapport à sa valeur finale de la porte en entrée et le passage à 50% de la transition par rapport à sa valeur finale de la porte en sortie (temps noté T_{p1} sur la figure 1.18) ;
– La seconde consiste à regarder la différence entre le temps de passage de 10% à 90% par rapport à sa valeur finale de la transition de la porte en sortie (ou de 90% à 10% dans le cas d'une transition opposée)(temps noté T_{p2} sur la figure 1.18)

C'est très souvent la première définition du temps de propagation qui est retenue, c'est d'ailleurs cette définition qui est utilisée dans les méthodes d'insertion de buffers. On trouve dans la littérature beaucoup de techniques ayant pour objectif de modéliser le temps de propagation d'une

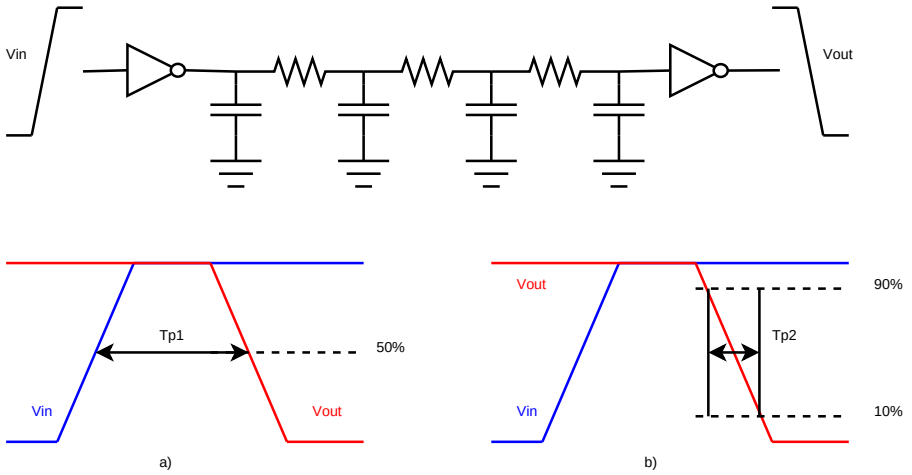


FIG. 1.18 – Les deux définitions (T_{p1} et T_{p2}) du temps de propagation sur une interconnexion. T_{p1} représente la différence de temps entre le passage à 50% de la transition par rapport à sa valeur finale de la porte en entrée et le passage à 50% de la transition par rapport à sa valeur finale de la porte en sortie. T_{p2} représente la différence entre le temps de passage de 10% à 90% par rapport à sa valeur finale de la transition de la porte en sortie (ou de 90% à 10% dans le cas d'une transition opposée).

porte *CMOS* pilotant une charge capacitive.

Il est possible en utilisant le schéma de la figure 1.19 d'obtenir une première approximation du temps de propagation au sein d'une interconnexion :

$$T1 = (R_{buffer} + R_{fil})\,(C_{IN} + C_{fil}) \tag{1.26}$$

- R_{buffer} représente la résistance équivalente de l'inverseur ;
- R_{fil} représente la résistance du fil ;
- C_{IN} représente la capacité d'entrée de l'inverseur en sortie de ligne ;
- C_{fil} représente la capacité totale présentée par le fil.

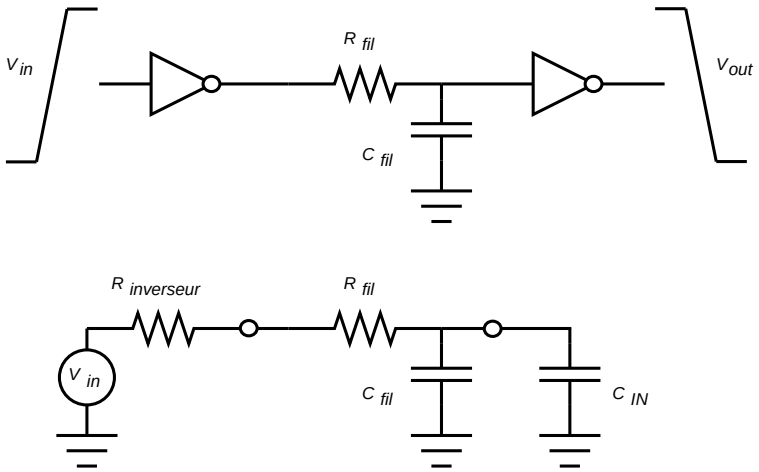


FIG. 1.19 – Schéma équivalent *lumped* d'une interconnexion avec ses buffers d'entrée et de sortie.

Tous ces paramètres ont été définis dans la section précédente. Cette première approximation n'utilise pas un modèle distribué pour l'interconnexion mais un modèle *lumped* ce qui diminue sa fiabilité (cf figure 1.5 sur l'allure de la réponse à une rampe des différents modèles physiques pour le fil).
La formule la plus utilisée pour le calcul du temps de propagation a été introduite par Elmore dans [Elm48]. Bien que cette formule ne considère toujours pas la notion de modèle distribué des interconnexions, c'est celle que les concepteurs utilisent afin d'obtenir une première estimation rapide et approximative du temps de propagation.

$$T2 = 0.5R_{fil}C_{fil} + R_{buffer}C_{fil} + R_{fil}C_{IN} + R_{buffer}C_{IN} + R_{fil}C_{fil} \tag{1.27}$$

On peut trouver dans [Sak93] une évolution de la formule présentée par [Elm48] qui prend en compte l'utilisation de modèles distribués pour les lignes d'interconnexion.

$$T3 = 0.1R_{fil}C_{fil} + ln\left(\frac{1}{1-v}\right)(R_{buffer}C_{fil} + R_{fil}C_{IN} + R_{buffer}C_{IN} + 0.4R_{fil}C_{fil}) \tag{1.28}$$

$T3$ représente le temps de propagation mis pour que la sortie atteigne une valeur normalisée (ici dans le cas de la définition première du temps de propagation on prendra pour valeur finale $\frac{V_{dd}}{2}$) avec $v = \frac{valeur}{V_{dd}}$ soit $v = 0.5$ ce qui donne en remplacant :

$$T3 = 0.377R_{fil}C_{fil} + 0.693\,(R_{buffer}C_{fil} + R_{fil}C_{IN} + R_{buffer}C_{IN}) \tag{1.29}$$

Méthodes d'insertion de buffer

Avec l'augmentation de la longueur des interconnexions, ainsi qu'avec l'évolution des paramètres technologiques, le temps de propagation au sein des interconnexions devient critique et par conséquent, il faut avoir recours à des méthodes d'insertion de buffers afin d'accélérer la propagation des signaux.

Bufferisation On-Chip Beaucoup de travaux ont été menés autour de l'insertion de buffers dans les interconnexions [BM85], [LCL95], [NB00],[NB01],[CF06],[LMHY05]. Nous allons présenter ici la méthode la plus utilisée [BM85] ainsi que ses dérivées [NB00],[NB01],[CF06],[LMHY05]. L'expression du premier ordre du temps de propagation sur un fil peut être définie de la manière suivante :

$$T_{fil} = R_{fil/m}C_{fil/m}L^2 \tag{1.30}$$

- $R_{fil/m}$ représente la résistance du fil par unité de longueur ;
- $C_{fil/m}$ représente la capacité du fil par unité de longueur ;
- L représente la longueur du fil.

Le temps de propagation au sein d'une interconnexion évolue en fonction du carré de la longueur du fil. Lorsque l'on insère des buffers, l'interconnexion est fractionnée en plusieurs segments de longueur $\frac{L}{K-1}$ où K représente le nombre de buffers insérés le long de la ligne tel que le montre

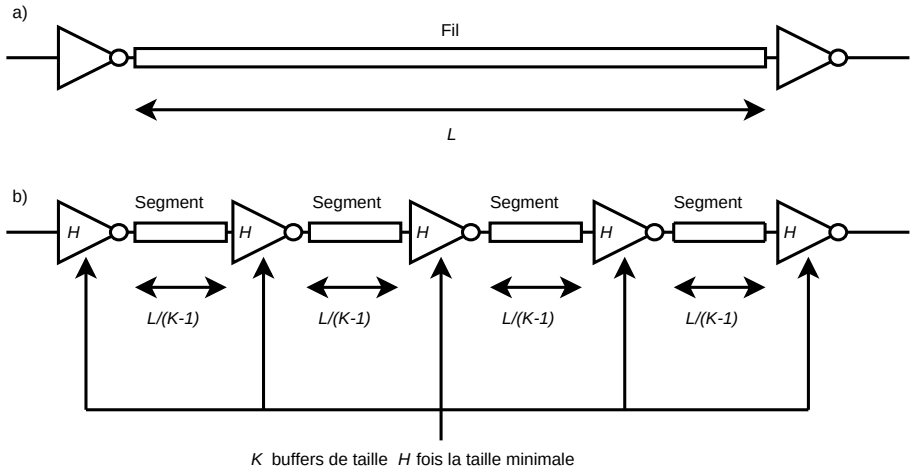


FIG. 1.20 – a) Interconnexion classiquement bufferisée, b) Interconnexion bufferisée complètement.

la figure 1.20 b). Le fait d'insérer des buffers va ici faire augmenter le temps de propagation dû aux portes puisqu'elles sont plus nombreuses que dans le cas d'une interconnexion non bufferisée. C'est pour cela que les méthodes d'insertion de buffers définissent un nombre optimal de buffers K_{opt} à insérer ainsi qu'une taille optimale de ces mêmes buffers H_{opt} afin d'obtenir le temps de propagation minimal sur le fil et donc les performances temporelles maximales.
En utilisant l'insertion de buffers, le temps de propagation sur le fil devient donc :

$$T_{fil} = \frac{R_{fil/m} C_{fil/m} L^2}{K} + (K-1) T_{buffer} \tag{1.31}$$

- K représente le nombre de buffers insérés sur la ligne ;
- T_{buffer} représente le temps de propagation à travers un buffer ; ce temps dépend de la charge vue par le buffer (ie : la capacité du fil ou du segment que le buffer doit piloter) ainsi que de la taille de ce dernier.

Soit en reprenant l'équation 1.29, on obtient :

$$T_{fil} = 0.377 \frac{R_{fil} C_{fil}}{K} + 0.693 \left(R_{buffer} C_{fil} + R_{fil} C_{IN} + K R_{buffer} C_{IN} \right) \tag{1.32}$$

Nous avons montré précédemment que la résistance de canal d'un transistor MOS est inversement proportionnelle au rapport $\frac{W_{MOS}}{L_{MOS}}$ alors que la capacité d'entrée d'un transistor MOS est, quant à elle, proportionnelle au rapport $\frac{W_{MOS}}{L_{MOS}}$. L_{MOS} étant fixe (Longueur du canal du transistor propre à la technologie employée), c'est la largeur du transistor W qui intervient dans la variation des paramètres résistifs et capacitifs. Nous pouvons donc définir les deux nouvelles équations en posant H comme étant la taille du buffer :

$$R_{buffer} = \frac{R_0}{H} \tag{1.33}$$

$$C_{IN} = C_0 H \tag{1.34}$$

	K_{opt}	H_{opt}
[BM85]	$\sqrt{\frac{0.377R_{fil}C_{fil}}{0.693R_0C_0}}$	$\sqrt{\frac{R_0C_{fil}}{R_{fil}C_0}}$
[NB01]	$\sqrt{\frac{A_1R_{fil}C_{fil}}{A_2}}$	$\sqrt{\frac{A_3C_{fil}}{R_{fil}A_4}}$
[CF06]	$\sqrt{\frac{A_5R_{fil}C_{fil}}{A_6R_0C_0}}$	$\sqrt{\frac{R_0C_{fil}}{R_{fil}C_0}}$
[LMHY05]	$\sqrt{\frac{A_7R_{fil}C_{fil}}{2R_0(C_{IN_0}+C_{OUT_0})}}$	$\sqrt{\frac{R_0C_{fil}}{R_{fil}C_0}}$

TAB. 1.6 – Expression des équations de K_{opt} et H_{opt} pour différents travaux où les A_i représentent des constantes technologiques dépendantes des paramètres des buffers.

- R_0 représente la résistance d'un buffer de taille minimale dans la technologie donnée ;
- C_0 représente la capacité d'entrée d'un buffer de taille minimale dans cette même technologie.

En substituant R_0 et C_0 dans l'équation 1.32, on obtient :

$$T_{fil} = 0.377\frac{R_{fil}C_{fil}}{K} + 0.693\left(\frac{R_0}{H}C_{fil} + R_{fil}C_0H + KR_{buffer}C_{IN}\right) \tag{1.35}$$

Pour résoudre le système d'équations, on pose $\frac{\delta T_{fil}}{K} = 0$ et $\frac{\delta T_{fil}}{H} = 0$, nous obtenons alors la taille H_{opt} et le nombre K_{opt} optimal de buffers qui permettent d'obtenir le temps de propagation le plus court sur le fil :

$$K_{opt} = \sqrt{\frac{0.377R_{fil}C_{fil}}{0.693R_0C_0}} \tag{1.36}$$

$$H_{opt} = \sqrt{\frac{R_0C_{fil}}{R_{fil}C_0}} \tag{1.37}$$

Cette méthode d'insertion de buffers présentée dans [BM85] permet de conclure sur trois points.

- Le nombre optimal de buffers K_{opt} dépend du ratio du délai du fil $R_{fil}C_{fil}$ par rapport au temps de traversée de la porte R_0C_0.
- La taille optimale des buffers H_{opt} ne dépend pas de la longueur du fil car le rapport $\frac{C_{fil}}{R_{fil}}$ reste constant quelle que soit la longueur du fil sur une même couche de métal.
- La taille optimale des buffers H_{opt} est dimensionnée de manière à ce que la résistance du buffer R_{buffer} soit égale à la résistance du segment R_{fil} que ce buffer va piloter.

D'autres travaux ont été menés sur ce thème [NB00],[NB01],[CF06],[LMHY05] en prenant pour point de départ les équations définies dans [Elm48]. Les résultats de ces travaux sont présentés dans le tableau 1.6 et sont similaires à ceux de [BM85]. Lors de la phase de simulation ces différentes techniques ont été testées ; elles donnent toutes des valeurs de K_{opt} et H_{opt} quasiment identiques. La méthode de [BM85] a donc été retenue pour les expérimentations.

Bufferisation Off-Chip : adaptation pyramidale Les méthodes d'insertion de buffer présentées dans le paragraphe précédent sont efficaces pour des bus On-Chip. Lorsqu'il s'agit maintenant

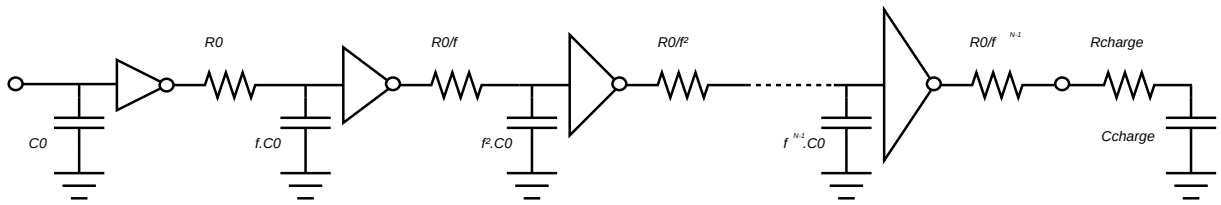


FIG. 1.21 – Schéma de l'adaptation pyramidale pour piloter une forte charge capacitive.

d'augmenter les performances avec comme contrainte une très forte charge capacitive (des pads de sortie d'une puce par exemple), ces techniques ne permettent pas d'atteindre de bonnes performances.
On utilisera alors préférentiellement la technique de l'adaptation pyramidale [Nem84]. Cette technique consiste en une mise en cascade de buffers avec pour particularité que chaque buffer inséré est de taille F fois supérieure au précédent tel que le montre la figure 1.21. L'expression du temps de propagation à travers la chaine d'inverseurs est alors donnée par la formule suivante :

$$T_{chaine} = 0.377 R_{charge} C_{charge} + 0.693 \left(\frac{R_0}{F^{N-1}} C_{charge} + (N-1) F R_0 C_0 + R_{charge} C_0 \right) \quad (1.38)$$

- R_0 représente la résistance d'un buffer de taille minimale dans la technologie donnée ;
- C_0 représente la capacité d'entrée d'un buffer de taille minimale dans cette même technologie ;
- R_{charge} représente la résistance du noeud à charger en sortie de la chaine d'inverseur ;
- C_{charge} représente la capacité du noeud à charger en sortie de la chaine d'inverseur.

Il a été démontré dans [LL75],[MC80] que le délai minimum dans la chaine d'inverseurs est obtenu pour un rapport des tailles des inverseurs exponentiel. Ces travaux proposent alors une détermination du rapport de taille optimal et du nombre optimal d'inverseurs de la chaîne.

$$F = \frac{W_i}{W_{i-1}} = e \quad (1.39)$$

- W_i représente la largeur des transistors du i^{eme} inverseur et respectivement le $i-1^{eme}$ pour W_{i-1} ;
- e est la base du logarithme.

$$N = \ln \left(\frac{C_{charge}}{C_0} \right) \quad (1.40)$$

Nos travaux de recherche concernant exclusivement la caractérisation des bus On-Chip, c'est la première méthode d'insertion de buffer introduite dans cette partie qui sera utilisée.
Nous disposons maintenant de tous les paramètres physiques qui permettent de dimensionner une interconnexion ainsi qu'un bus. Nous disposons également des paramètres physiques des buffers, nous pouvons donc maintenant modéliser la consommation d'un bus.

1.4 Bilan

Ce chapitre a permis de présenter la difficulté d'une modélisation précise au niveau physique des différents éléments qui constituent un bus. Les problèmes de consommation et de délai dûs tout particulièrement aux évolutions des paramètres technologiques ont été présentés.

Chapitre 2

Estimation de la consommation des interconnexions

Sommaire

Résumé

Dans le second chapitre, la méthode d'estimation de la consommation des interconnexions est proposée. Suite à la modélisation du bus au niveau technologique, les paramètres importants intervenant dans la variation de la consommation (technologie, couche de métal, longueur de bus...) ont été extraits. Des simulations SPICE de ces circuits ont été réalisées ; les résultats expérimentaux ont permis d'obtenir des tableaux multi-entrées inclus au sein d'un outil d'estimation. Cet outil (*Interconnect Explorer*) permet alors à l'utilisateur, après configuration, (i.e. choix de la technologie, de la couche de métal, de la longueur de bus) d'obtenir très rapidement une estimation de la consommation du transfert de données sur un bus. Les expérimentations de validation montrent que l'outil permet d'obtenir une estimation avec une erreur maximale de 3% (par rapport aux simulations SPICE) avec un temps d'exécution de quelques secondes (une simulation SPICE dans les mêmes conditions expérimentales prenant plusieurs heures).

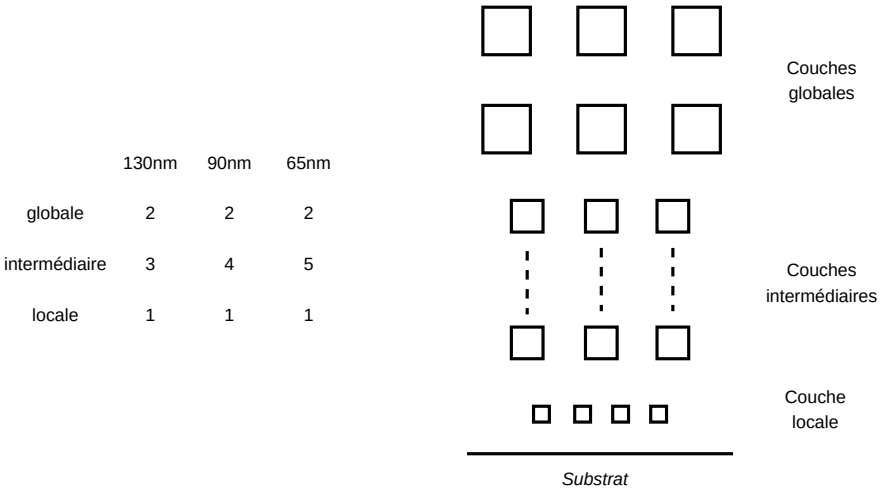


FIG. 2.1 – Répartition des couches de métal.

2.1 Extraction des paramètres impactant la consommation

L'objectif de la modélisation de la consommation des interconnexions est d'obtenir un modèle mathématique dont les paramètres d'entrée seront choisis par l'utilisateur et de fournir des résultats rapides et précis en termes de consommation et de délai.
Le second objectif de la modélisation est de permettre le test rapide de l'efficacité des techniques d'optimisation des performances proposées dans la littérature et également de tester celles que nous allons proposer.
Afin d'effectuer les expérimentations, il est indispensable, dans un premier temps, d'identifier les différents paramètres qui vont faire évoluer la consommation.
Puisque nous avons choisi d'effectuer la modélisation au niveau physique, le premier paramètre qui rentre en compte dans la modélisation est le choix de la technologie. Nous allons effectuer les expérimentations pour trois technologies : 130, 90 et 65nm.
Chaque technologie dispose d'un nombre de couches de métal qui lui est propre ; un second paramètre sera donc le niveau de métal sur lequel sera placé le bus. Chaque niveau de métal a une finalité différente et des propriétés physiques (dimensions) différentes :

- la couche de métal la plus basse, de section très faible, sera réservée pour de courtes interconnexions au sein même des cellules (couche locale) ;
- la couche de métal immédiatement supérieure sera quant à elle réservée pour les bus locaux à l'intérieur de ces cellules (couche intermédiaire) ;
- les couches de métal supérieures seront réservées pour les bus entre les cellules ; ce sont ces couches de métal qui vont nous intéresser plus particulierement car c'est sur celles-ci que seront placés les bus (couche intermédiaire) ;
- les deux dernières couches de métal, les plus hautes (couches globales), qui ont la section la plus importante seront réservées à la distribution de l'arbre d'horloge, aux lignes d'alimentation ainsi qu'aux bus globaux (les bus les plus longs).

La figure 2.1 présente, pour les trois technologies qui ont été modélisées, la répartition des couches de métal. Nous avons montré précédemment que la capacité présentée par le fil va varier en fonction

Paramètre	Type	Plage de variation	Nombre de valeurs
Technologie	Technologique	130nm / 90nm / 65nm	3
Couche de métal	Technologique	1 à 6 (130nm)	6
		1 à 7 (90nm)	7
		1 à 8 (65nm)	8
Longueur	Architectural	$0 < L \leqslant 10mm$	9 $L \in [0.1, 0.3, 0.5, 0.7, 1, 3, 5, 7, 10]$
Bufferisation	Technologique	Simple / Complète	2
Transition	Algorithmique	Combinaison	22

TAB. 2.1 – Paramètres entrant en jeu dans la modélisation de la consommation.

de la couche de métal choisie. En effet, les dimensions (épaisseur, hauteur, largeur, espacement) diffèrent en fonction de la couche sur laquelle se trouve le bus donc les capacités parasites (fil par rapport au substrat et *crosstalk*) qui interviennent dans l'équation de la consommation varient également.

Le chapitre précédent a expliqué que lorsqu'une ligne d'interconnexion devient longue, nous sommes alors confrontés à un problème de temps de propagation, problème que nous pouvons résoudre par l'insertion de buffers. Donc, la longueur de l'interconnexion ainsi que l'insertion de buffers seront des paramètres pour les expérimentations.

Dans la section sur le *crosstalk*, nous avons vu (tableau 1.4) que la capacité de *crosstalk* pouvait avoir une influence plus ou moins importante sur le retard de propagation en fonction du type de transition observée sur le fil victime. Cette modification de la capacité de *crosstalk* va également agir sur la consommation. Chaque type de transition du tableau 1.4 sera donc un paramètre pour les expérimentations.

Remarque : dans le tableau 1.4, un fil peut rester à un niveau stable entre deux cycles d'horloge i.e. pas de transition. Pour le cas où il n'y a pas de transition, les expérimentations ont été effectuées pour les 2 niveaux logiques stables que sont V_{dd} et GND.

Nous avons également simulé les cas où le fil victime reste à un niveau stable V_{dd} ou GND.

Pour résumer, chaque paramètre, sa plage de variation, son type ainsi que le nombre de valeurs qu'il peut prendre sont représentés dans le tableau 2.1. En utilisant ces paramètres ainsi que le modèle physique du bus défini précédemment, la modélisation du délai et de la consommation a été réalisée au niveau transistor en utilisant un simulateur SPICE (ELDO v5.7). Afin de réaliser les 8316 simulations (6 (couches de métal en 130nm) x 9 (longueurs de bus) x 2 (types de bufferisation) x 22 (combinaisons de transition) + 7 (couches de métal en 90nm) x 9 (longueurs de bus) x 2 (types de bufferisation) x 22 (combinaisons de transition) + 8 (couches de métal en 65nm) x 9 (longueurs de bus) x 2 (types de bufferisation) x 22 (combinaisons de transition) = 8316 circuits), un générateur automatique des netlists représentant les circuits a été conçu. Un exemple de netlist générée pour la simulation d'un circuit SPICE est donné en tableau 2.2. La représentation sous forme du circuit associé est illustrée sur la figure 2.2. Les différentes étapes de construction du circuit sont représentées dans la netlist et sur la figure associée. De plus, en fin de listing, il est possible de positionner des instructions spécifiques permettant d'extraire des valeurs de variation d'une grandeur choisie (i.e. variation de tension, de courant, de temps de propagation ...) en un

nœud précis du circuit. En ce qui nous concerne, nous extrayons les valeurs de consommation énergétique et de temps de propagation sur le fil victime pour chacun des 8316 circuits à simuler. Les résultats de simulation en termes de délai et de consommation énergétique pour chacun des circuits simulés ont permis d'établir des tableaux multi-entrées dont les entrées sont les paramètres définis dans le tableau 2.1. La combinaison de chacune des valeurs des paramètres d'entrée permet de pointer sur une valeur de consommation énergétique et de temps de propagation. Ces tableaux ont été inclus au sein d'un outil –*Interconnect Explorer* (présenté dans [CSLJ08b])– développé au cours de cette thèse. Nous allons dans la section suivante expliquer le fonctionnement de cet outil.

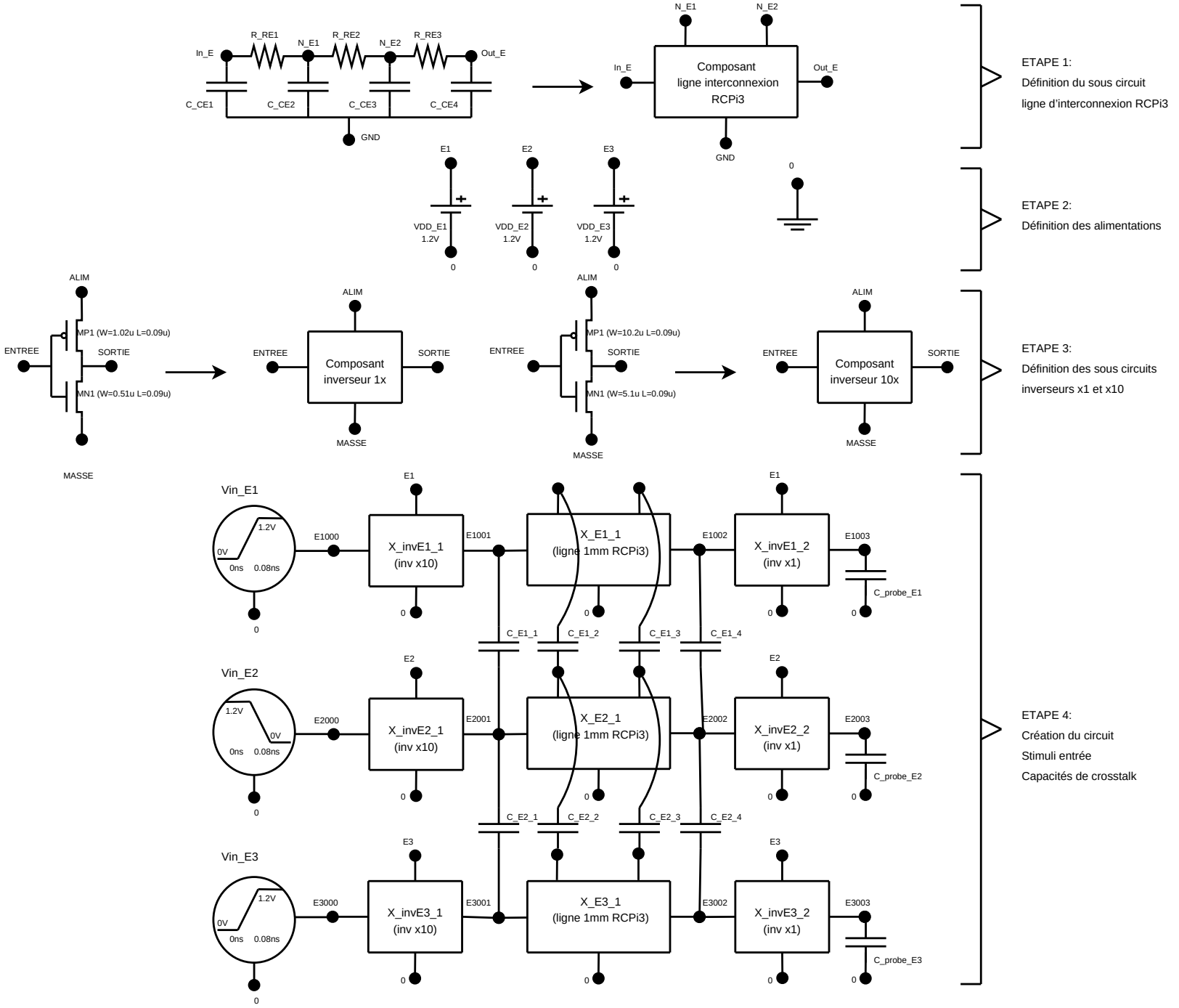


FIG. 2.2 – Etapes de construction d'un circuit pour la simulation SPICE (représentation circuit).

```
*-------------------------------------------------------------------------------------
*| Exemple de Simulation SPICE: 90nm / ML 2 / 1mm / FO4 / 101 transition pattern     |
*-------------------------------------------------------------------------------------
.lib Lib90nm.lib TT
.option nomod notrc lcapop hmin=1ps hmax=10ps smooth noasc aex
.opt
*-------------------------------------------------------------------------------------
*|ETAPE 1: Définition du sous circuit ligne d'interconnexion RCPi3                   |
*-------------------------------------------------------------------------------------
.SUBCKT ligne1.0mm GND In_E Out_E
R_RE1    In_E      N_E1      81.4814853516
R_RE2    N_E1      N_E2      81.4814853516
R_RE3    N_E2      Out_E     81.4814853516
C_CE1    GND       In_E      5.7037039746e-15
C_CE2    GND       N_E1      11.4074079492e-15
C_CE3    GND       N_E2      11.4074079492e-15
C_CE4    GND       Out_E     5.7037039746e-15
.ENDS ligne1.0mm

*-------------------------------------------------------------------------------------
*| ETAPE 2: Définition des alimentations                                             |
*-------------------------------------------------------------------------------------
.PARAM Signal = 1.2
VDD_E1 E1 0 DC Signal
VDD_E2 E2 0 DC Signal
VDD_E3 E3 0 DC Signal

*-------------------------------------------------------------------------------------
*| ETAPE 3: Définition des sous circuits inverseurs                                  |
*-------------------------------------------------------------------------------------
.SUBCKT inv_str1 ALIM MASSE ENTREE SORTIE                  .SUBCKT inv_str10 ALIM MASSE ENTREE SORTIE
MN1 MASSE ENTREE SORTIE MASSE N  W= 0.51U  L= 0.090U       MN1 MASSE ENTREE SORTIE MASSE N  W= 5.10U   L= 0.090U
MP1 ALIM  ENTREE SORTIE ALIM  P  W= 1.02U  L= 0.090U       MP1 ALIM  ENTREE SORTIE ALIM  P  W= 10.20U  L= 0.090U
.ENDS inv_str1                                             .ENDS inv_str10

*-------------------------------------------------------------------------------------
*| ETAPE 4: Creation du circuit, des stimuli, intégration des capacités de crosstalk |
*-------------------------------------------------------------------------------------
Vin_E1 E1000 0 pwl (0n 0.0 0.08n 1.2)    Vin_E2 E2000 0 pwl (0n 1.2 0.08n 0.0)    Vin_E3 E3000 0 pwl (0n 0.0 0.08n 1.2)
X_invE1_1 E1 0 E1000 E1001 inv_str10     X_invE2_1 E2 0 E2000 E2001 inv_str10     X_invE3_1 E3 0 E3000 E3001 inv_str10
X_E1_1 0 E1001 E1002 Ligne1.0mm          X_E2_1 0 E2001 E2002 Ligne1.0mm          X_E3_1 0 E3001 E3002 Ligne1.0mm
X_invE1_2 E1 0 E1002 E1003 inv_str1      X_invE2_2 E2 0 E2002 E2003 inv_str1      X_invE3_2 E3 0 E3002 E3003 inv_str1
C_probe_E1 E1003 0 5f                    C_probe_E2 E2003 0 5f                    C_probe_E3 E3003 0 5f

*-------------------------------------------------------------------------------------
*| couplage crosstalk                                          |
*-------------------------------------------------------------------------------------
C_E1_1  X_E1_1.In_E  X_E2_1.In_E  21.0000009974e-15    C_E2_1  X_E2_1.In_E  X_E3_1.In_E  21.0000009974e-15
C_E1_2  X_E1_1.N_E1  X_E2_1.N_E1  10.5000004987e-15    C_E2_2  X_E2_1.N_E1  X_E3_1.N_E1  10.5000004987e-15
C_E1_3  X_E1_1.N_E2  X_E2_1.N_E2  10.5000004987e-15    C_E2_3  X_E2_1.N_E2  X_E3_1.N_E2  10.5000004987e-15
C_E1_4  X_E1_1.Out_E X_E2_1.Out_E 21.0000009974e-15    C_E2_4  X_E2_1.Out_E X_E3_1.Out_E 21.0000009974e-15

*-------------------------------------------------------------------------------------
*| Extraction ENERGIE et TEMPS DE PROPAGATION                                        |
*-------------------------------------------------------------------------------------
.defwave positiveV_E=(abs(i(VDD_E2))-i(VDD_E2))*0.5*1.2
.tran .01n 10n
.extract integ(w(positiveV_E))
.extract TPD (v(E2000),v(E2003),VTH=0.60)
```

TAB. 2.2 – Etapes de construction d'un circuit pour la simulation SPICE (représentation netlist).

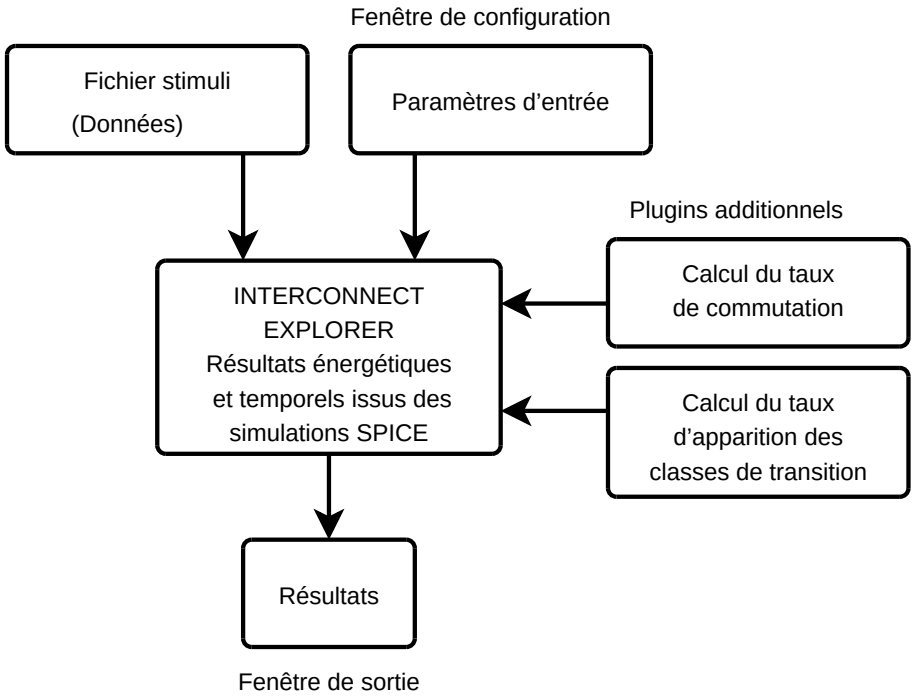


FIG. 2.3 – Flot d'estimation représentant le fonctionnement de l'outil.

2.2 Fonctionnement d'*Interconnect Explorer* : Flot d'estimation

Le flot d'estimation d'*Interconnect Explorer* est représenté en figure 2.3. Lors de l'utilisation d'*Interconnect Explorer*, l'utilisateur doit renseigner l'outil dans la fenêtre de configuration, en faisant des choix sur :

- la technologie ;
- la couche de métal ;
- la longueur du bus ;
- la largeur du bus ;
- la fréquence de fonctionnement ;
- le type de bufferisation ;
- le fichier de stimuli. Ce fichier de stimuli contient les données de son application : données qui vont circuler sur le bus (image, musique, instructions, adresses ...). Si l'utilisateur ne dispose pas de ce fichier, il a la possibilité de choisir dans l'outil un profil de données avec lequel l'estimation peut être effectuée. Les trois types de profils qui lui sont proposés sont les suivants :
  - un profil de données aléatoires (activité de 50% sur chacun des bits du bus). C'est avec ce profil qu'il peut obtenir la borne maximale de consommation pour le bus considéré ;
  - un profil de type données : transfert d'une image sur le bus (il est important de noter que le profil d'activité pour de la musique ou de la parole est rigoureusement identique) ;
  - un profil de type addresses (séquence d'adresses consécutives).

Les paramètres sont entrés à l'outil via l'interface graphique présentée en figure 2.4.

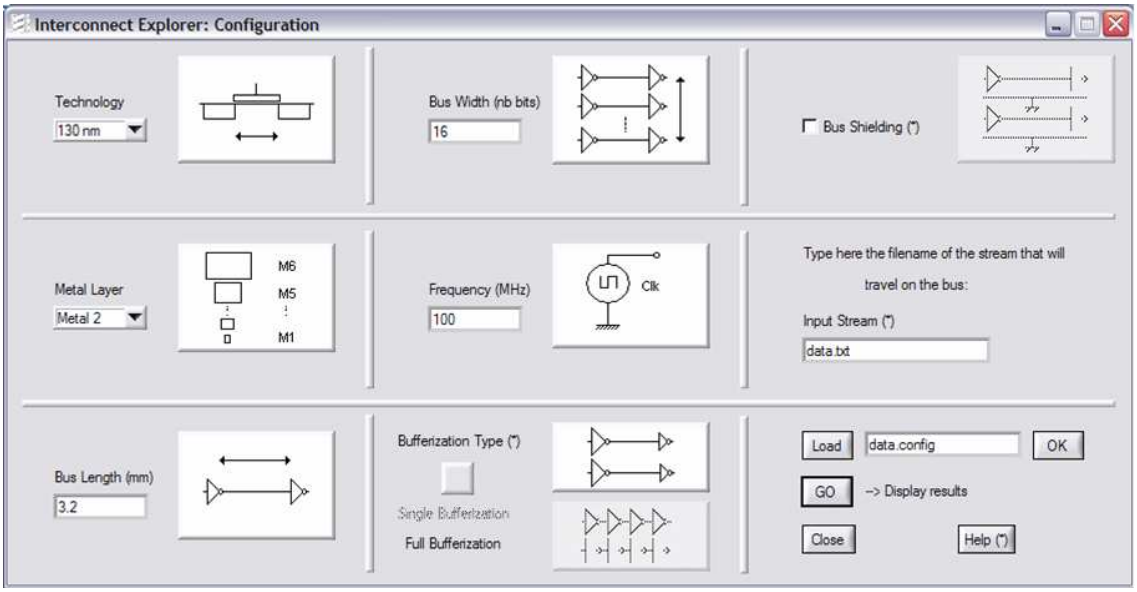


FIG. 2.4 – Interface d'entrée de l'outil.

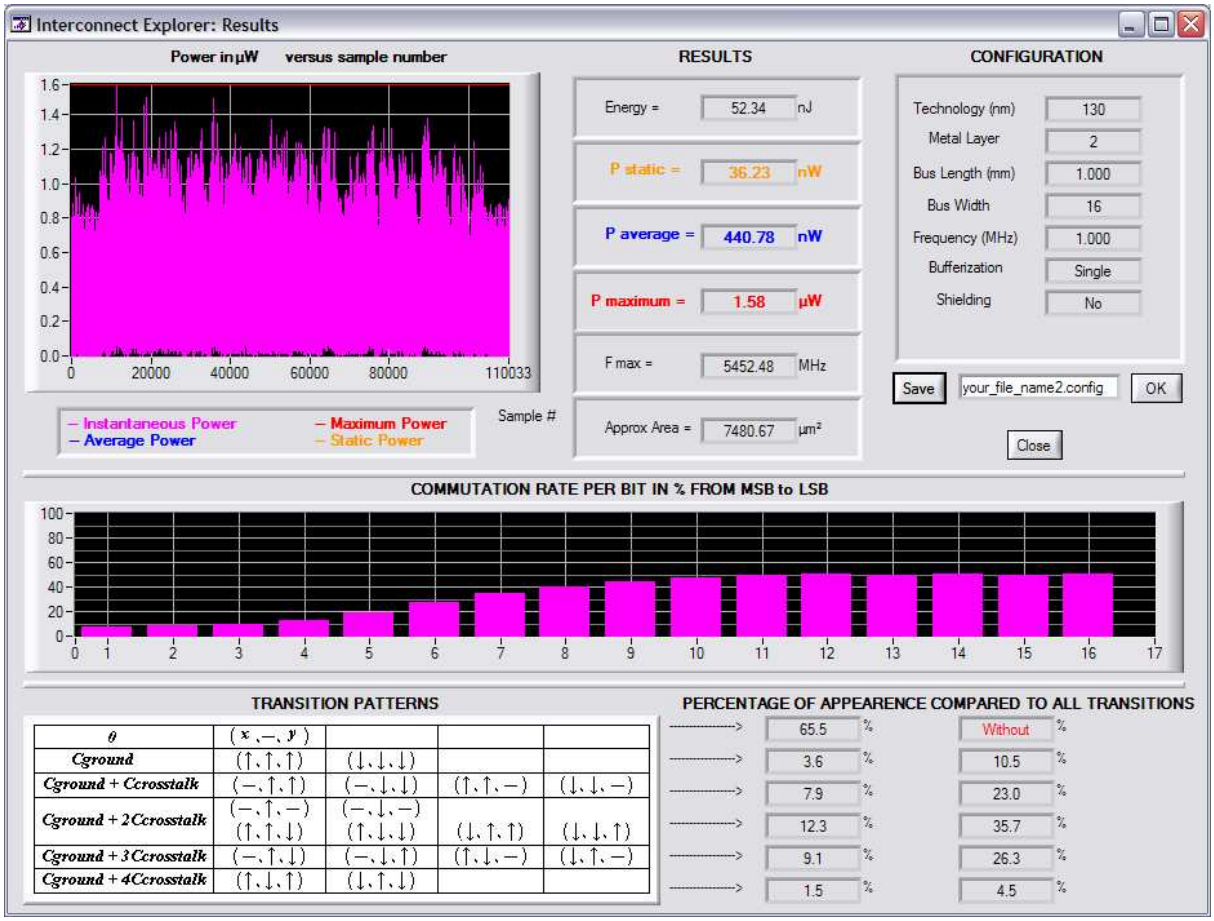


FIG. 2.5 – Interface de sortie de l'outil.

Une fois la configuration effectuée, *Interconnect Explorer* fournit en retour à l'utilisateur, dans la fenêtre de sortie (figure 2.5), des informations sur :

– la consommation énergétique ; cette consommation est calculée en effectuant la double somme, sur le nombre de bits du bus (Nb_{bits}) et le nombre d'échantillons (Nb_{ech}), de la consommation énergétique de chaque transition, sur chaque fil i (avec $i \in [0, Nb_{bits}]$) et pour chaque donnée (ou échantillon) j (avec $j \in [0, Nb_{ech}]$) :

$$Energie_{totale} = \sum_{j=0}^{Nb_{ech}} \sum_{i=0}^{Nb_{bits}} Energie_{Transition(k)-Fil(i)-Echantillon(j)} \tag{2.1}$$

Il existe 22 sortes de transitions comme défini dans le tableau 2.1, répertoriées en plusieurs classes k, $k \in [0, 1, 1+r, 1+2r, 1+3r, 1+4r]$. La classe de la transition k est d'abord extraite en fonction des commutations sur le fil victime et ses deux voisins, puis la valeur de la consommation énergétique d'une transition appartenant à une classe k (i.e. $Energie_{Transition(k)}$) est retournée. Cette dernière est fonction de la technologie, de la couche de métal, de la longueur du bus et du type de bufferisation (i.e. valeur retournée des tableaux de résultats multi-entrées) et est issue de la simulation de l'un des circuits au niveau transistor. Dans la suite de cette section, chaque occurence du terme $Transition(k)$ signifie que l'on extrait la transition (i.e. chaque classe k possède un certain nombre de type de transition comme référencé dans le tableau 1.4) de la classe de transition k qui a eu lieu sur le fil i dans l'échantillon j.

– la consommation de puissance statique ; cette consommation est calculée en effectuant la double somme, sur le nombre de bits du bus et le nombre d'échantillons, de la consommation énergétique d'une transition de classe $k = 0$ (i.e. pas de changement d'état), sur chaque fil et pour chaque donnée (ou échantillon), par rapport au temps de cycle (i.e. période d'horloge) :

$$P_{statique} = \sum_{j=0}^{Nb_{ech}} \sum_{i=0}^{Nb_{bits}} Energie_{Transition(0)-Fil(i)-Echantillon(j)} \frac{1}{T_{cycle}} \tag{2.2}$$

– la consommation de puissance moyenne ; cette consommation est calculée en effectuant la double somme, sur le nombre de bits du bus et le nombre d'échantillons de la consommation énergétique de chaque transition, sur chaque fil et pour chaque donnée (ou échantillon), par rapport au temps de cycle, par rapport au nombre total de transitions (i.e. $Nb_{bits}.Nb_{ech}$) :

$$P_{moyenne} = \frac{\sum_{j=0}^{Nb_{ech}} \sum_{i=0}^{Nb_{bits}} Energie_{Transition(k)-Fil(i)-Echantillon(j)} \frac{1}{T_{cycle}}}{Nb_{ech} Nb_{bits}} \tag{2.3}$$

– la consommation de puissance maximale ; cette consommation est calculée en recherchant le maximum sur toutes les valeurs de la puissance instantanée :

$$P_{maximum} = \max\left(P_{instantanee(i,j)}\right) \tag{2.4}$$

La puissance instantanée d'une transition sur un fil est calculée en effectuant le rapport de la consommation énergétique de cette transition sur ce fil dans l'échantillon courant, par rapport au temps de cycle :

$$P_{instantanee(i,j)} = Energie_{Transition(k)-Fil(i)-Echantillon(j)} \frac{1}{T_{cycle}} \quad \forall i \in [0, Nb_{bits}] \text{ et} j \in [0, Nb_{ech}] \tag{2.5}$$

– la fréquence de fonctionnement maximale ; cette fréquence (déterminée par les transitions pire cas en temporel) est calculée en effectuant l'inverse du maximum sur toutes les valeurs du temps de propagation :

$$F_{maximum} = \frac{1}{\max\left(TempsPropagation_{Transition(k)-Fil(i)-Echantillon(j)}\right)} \quad \forall i \in [0, Nb_{bits}] \text{ et} j \in [0, Nb_{ech}] \tag{2.6}$$

– la surface occupée sur la puce (lignes d'interconnexion plus buffers) ; cette surface est calculée en effectuant la somme de la surface occupée par les fils et de la surface occupée par les buffers :

$$Surface = [Nb_{bits}W + (Nb_{bits} - 1)\, S]\, L + Nb_{bits}\, [S_{buffer_{driver}} + S_{buffer_{receiver}}] \quad \text{dans le cadre d'une bufferisation simple} \tag{2.7}$$

W, S, L, $S_{buffer_{driver}}$ et $S_{buffer_{receiver}}$ représentent respectivement, la largeur du fil, l'espacement entre deux fils, la longueur du fil, la surface occupée par le buffer driver d'une ligne et la surface occupée par le buffer receiver de la ligne. Ces grandeurs sont dépendantes de la technologie employée.

$$Surface = Nb_{seg}\, [Nb_{bits}W + (Nb_{bits} - 1)\, S]\, L_{seg} + [Nb_{seg} + 1]\, Nb_{bits} S_{buffer} \quad \text{dans le cadre d'une bufferisation complète} \tag{2.8}$$

L_{seg}, Nb_{seg} et S_{buffer} représentent respectivement, la longueur du segment de fil, le nombre de segment de fil et la surface occupée par un buffer utilisé dans le cadre d'une bufferisation complète. Ces grandeurs sont également dépendantes de la technologie employée.

Des "plug-ins" additionnels ont été inclus à l'outil afin de calculer le taux de commutation de chaque fil du bus ainsi que le pourcentage d'apparition de chaque classe de transition du tableau 1.4. Les résultats fournis par l'exécution de ces "plug-ins" s'obtiennent en analysant le flux de données du fichier de stimuli fourni pour la simulation.

– Le taux de commutation de chaque fil du bus est obtenu en faisant le ratio du nombre de changements d'état sur chaque fil (i.e. 0 vers 1 et 1 vers 0), par rapport au nombre

Cycle	$Bit2$	$Bit1$	$Bit0$
$T = t_1$	1	0	1
$T = t_2$	0	1	0
$T = t_3$	1	0	0
$T = t_4$	0	0	1
$T = t_5$	1	1	0
$T = t_6$	0	0	0
$T = t_7$	1	1	1
$T = t_8$	1	1	1

	$Bit2$	$Bit1$	$Bit0$
De t_1 vers t_2	1 changement d'état	1 changement d'état	1 changement d'état
De t_2 vers t_3	1 changement d'état	1 changement d'état	0 changement d'état
De t_3 vers t_4	1 changement d'état	0 changement d'état	1 changement d'état
De t_4 vers t_5	1 changement d'état	1 changement d'état	1 changement d'état
De t_5 vers t_6	1 changement d'état	1 changement d'état	0 changement d'état
De t_6 vers t_7	1 changement d'état	1 changement d'état	1 changement d'état
De t_7 vers t_8	0 changement d'état	0 changement d'état	0 changement d'état
Total	6 changements d'état	5 changements d'état	4 changements d'état
Activité	6/8	5/8	4/8

TAB. 2.3 – Illustration du calcul du taux d'activité de chacun des bits du bus.

d'échantillons transmis.

$$Taux_{commutation(i)} = \frac{\sum_{j=0}^{Nb_{ech}} Changements \begin{pmatrix} 0 \rightarrow 1 \\ 1 \rightarrow 0 \end{pmatrix}}{Nb_{ech}} \quad \forall i \in [0, Nb_{bits}] \tag{2.9}$$

Le tableau 2.3 illustre un exemple du calcul du taux de commutation sur chaque fil d'un bus de 3 bits. Le taux de commutation sera utile par la suite pour tester les techniques d'optimisation des performances.

– Le taux d'apparition de chaque classe de transition est obtenu en faisant le ratio du nombre de transitions observées dans chacune des classes par rapport au nombre total de transitions.

$$Taux_{Classe(k)} = \frac{\sum_{i=0}^{Nb_{bits}} Transition_{Classe(k)}}{Nb_{ech} Nb_{bits}} \quad \forall j \in [0, Nb_{ech}] \text{ pour } k \in [0, 1, 1+r, 1+2r, 1+3r, 1+4r] \tag{2.10}$$

Le tableau 2.4 illustre un exemple du calcul du taux d'apparition de chacune des classes de transition sur un bus de 3 bits. Pour cela, il faut toujours considérer un fil victime et ses deux voisins puis se décaler itérativement sur chaque fil jusqu'à avoir parcouru tous les fils du bus. Pour les fils de poids fort et de poids faible du bus qui n'ont qu'un seul voisin, nous considérons que le second voisin (virtuel, noté $Bit_{V_{MSB}}$ pour le voisin du bit de poids fort du bus et respectivement $Bit_{V_{LSB}}$ pour le voisin du bit de poids faible du bus) reste à un

Cycle	$Bit_{V_{MSB}}$	$Bit2$	$Bit1$	$Bit0$	$Bit_{V_{LSB}}$
$T = t_1$	0	1	0	1	0
$T = t_2$	0	0	1	0	0
$T = t_3$	0	1	0	0	0
$T = t_4$	0	0	0	1	0
$T = t_5$	0	1	1	0	0
$T = t_6$	0	0	0	0	0
$T = t_7$	0	1	1	1	0
$T = t_8$	0	1	1	1	0

	$Bit2$		$Bit1$		$Bit0$	
De t_1 vers t_2	$(-,\downarrow,\uparrow)$	$1+3r$	$(\downarrow,\uparrow,\downarrow)$	$1+4r$	$(\uparrow,\downarrow,-)$	$1+3r$
De t_2 vers t_3	$(-,\uparrow,\downarrow)$	$1+3r$	$(\uparrow,\downarrow,-)$	$1+3r$	$(\downarrow,-,-)$	0
De t_3 vers t_4	$(-,\downarrow,-)$	$1+2r$	$(\downarrow,-,\uparrow)$	0	$(-,\uparrow,-)$	$1+2r$
De t_4 vers t_5	$(-,\uparrow,\uparrow)$	$1+r$	$(-,-,\downarrow)$	0	$(\uparrow,\downarrow,-)$	$1+3r$
De t_5 vers t_6	$(-,\downarrow,\downarrow)$	$1+r$	$(\downarrow,\downarrow,-)$	$1+r$	$(\downarrow,-,-)$	0
De t_6 vers t_7	$(-,\uparrow,\uparrow)$	$1+r$	$(\uparrow,\uparrow,\uparrow)$	1	$(\uparrow,\uparrow,-)$	$1+r$
De t_7 vers t_8	$(-,-,-)$	0	$(-,-,-)$	0	$(-,-,-)$	0

Classe	0	1	$1+r$	$1+2r$	$1+3r$	$1+4r$
Occurence	7/21	1/21	5/21	2/21	5/21	1/21

TAB. 2.4 – Illustration du calcul du taux d'apparition de chaque classe de transition.

niveau logique stable. Le taux d'apparition de chaque classe de transition sera utile par la suite pour tester les techniques d'optimisation des performances.

2.3 Validation d'*Interconnect Explorer*

Dans le but de vérifier la précision de l'outil, diverses comparaisons ont été effectuées entre des simulations SPICE (considérées comme la référence) et les résultats fournis par *Interconnect Explorer* comme illustré sur la figure 2.6.
Afin d'utiliser toute la plage de variation de l'outil, l'ensemble des possibilités de configuration de l'outil a été balayé. La figure 2.6 représente pour deux types de stimuli (image et données aléatoires) la précision, le temps d'exécution et la taille des fichiers manipulés d'*Interconnect Explorer* par rapport aux simulations SPICE.

2.3.1 Précision

Les graphiques a) et d) de la figure 2.6 représentent la précision de l'outil pour deux types de fichier de stimuli (image et données aléatoires) en fonction du nombre d'échantillons. Nous pouvons remarquer que la précision de l'outil augmente avec le nombre d'échantillons pris et converge rapidement vers une erreur faible. Sur ces graphiques, les estimations de consommation fournies par l'outil présentent une différence de seulement 3.1% pour un flot de type image et de 1.2% pour un flot de données aléatoires par rapport aux simulations SPICE. Il est donc possible, si l'on veut réduire la taille des fichiers de stimuli à stocker de ne conserver qu'une proportion faible du nombre d'échantillons total car les résultats convergent très rapidement.

2.3.2 Temps d'exécution

Les graphiques b) et e) de la figure 2.6 représentent les temps de calcul pour obtenir les résultats de consommation. L'outil permet d'obtenir ces résultats rapidement (environ 1 seconde) alors que les simulations SPICE peuvent durer jusqu'à plusieurs heures selon le volume de données à traiter.
Remarque : le fichier de stimuli de type image contient environ 600000 échantillons (i.e. 200000 pixels répartis dans les trois composantes de couleur RVB), la simulation complète du transfert de l'image sur un bus avec le simulateur SPICE peut donc prendre jusqu'à 80 heures environ (i.e. aux alentours de 5000 secondes pour 10000 échantillons, voir graphique b) de la figure 2.6) alors que *Interconnect Explorer* permet l'obtention du résultat en quelques secondes.

2.3.3 Taille de fichier

Enfin, les graphiques c) et f) de la figure 2.6 représentent la taille des fichiers manipulés par l'outil et par le simulateur SPICE. Nous remarquons que la taille des fichiers manipulés par l'outil est beaucoup plus petite et, de plus, croît beaucoup plus lentement que celle des fichiers manipulés par le simulateur SPICE. Il existe un rapport d'environ 25 entre la taille des fichiers manipulés par le simulateur SPICE et ceux manipulés par *Interconnect Explorer*.

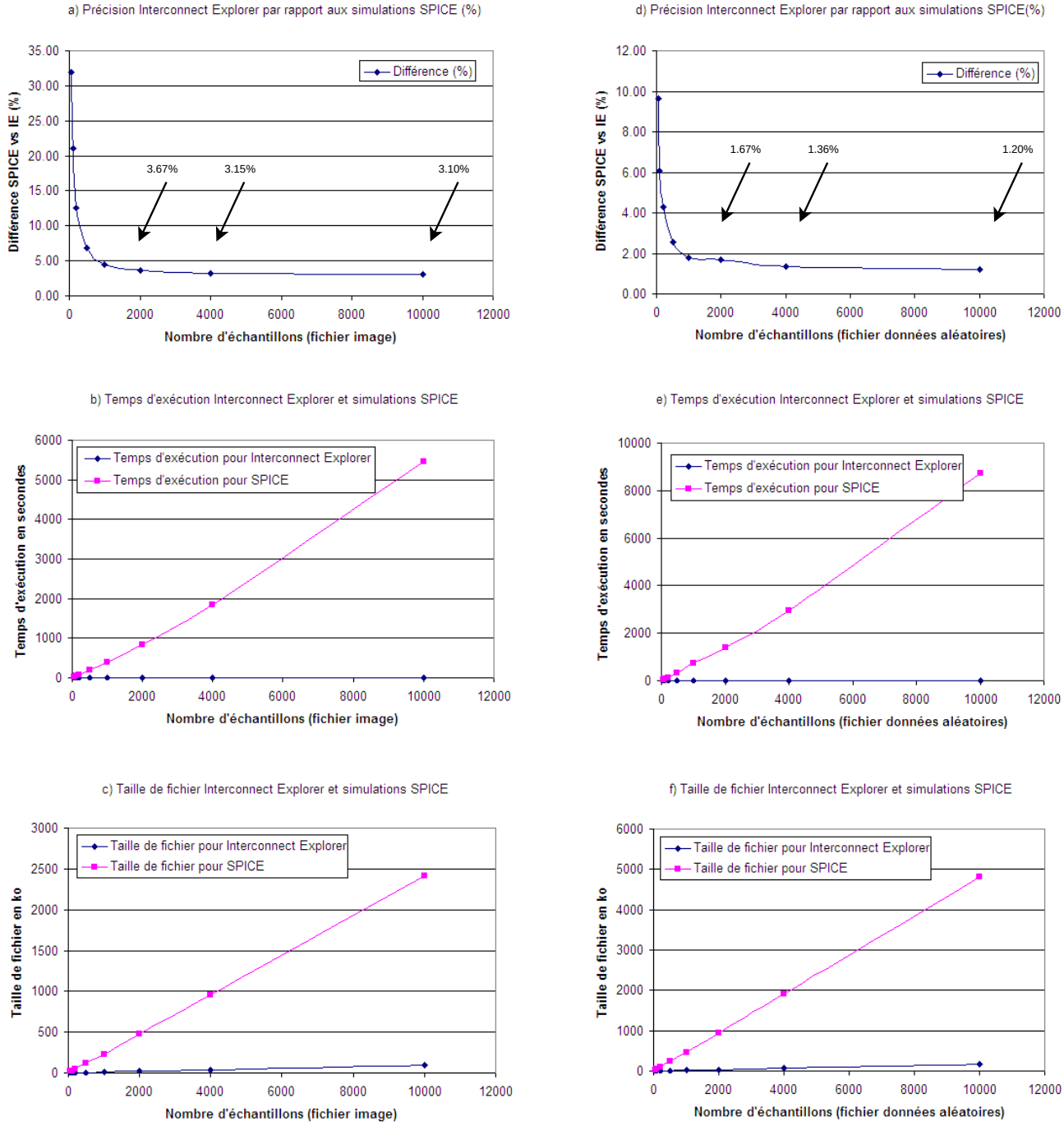


FIG. 2.6 – Comparaison entre *Interconnect Explorer* et SPICE en termes de précision, de temps d'exécution et de taille de fichier.

2.4 Bilan

Dans un premier temps, nous avons présenté dans ce chapitre l'extraction des paramètres impactant la consommation d'un bus. Puis la caractérisation et l'obtention des modèles par le biais de simulations SPICE ont été abordés.
Les expérimentations ont permis d'aboutir à un outil d'estimation (*Interconnect Explorer*), qui nous permettra de tester et de conclure sur l'efficacité des différentes techniques d'optimisation de la consommation et de délai présentées dans le prochain chapitre. Nous verrons également qu'il va nous permettre de définir de nouvelles voies d'optimisation.
Le prochain chapitre s'intéressera aux différentes techniques d'optimisation des performances existantes pour, dans un premier temps, en présenter un état de l'art, puis une synthèse, pour finalement en extraire les pistes à suivre afin de proposer dans le chapitre suivant des techniques d'optimisation innovantes et efficaces au niveau architectural.

Chapitre 3

Etat de l'art sur les techniques d'optimisation des performances

Sommaire

Résumé

Dans ce troisième chapitre, un état de l'art des principales techniques d'optimisation de la consommation et du délai est présenté. Dans un premier temps, l'outil d'estimation présenté dans le chapitre précédent nous permet de valider l'efficacité de ces techniques sur les paramètres impactant la consommation (activité, temps de propagation, capacités parasites...).
Dans un second temps, l'analyse des résultats fournis par l'outil permet de montrer que les techniques d'optimisation n'agissent pas forcément sur les bons paramètres afin de réduire la consommation.
A la fin de ce chapitre, de nouvelles pistes d'optimisation, en adéquation avec les résultats précédents, sont proposées.

3.1 Techniques d'optimisation des performances (temps et consommation)

Comme nous l'avons vu dans l'équation de la puissance dynamique (cf équation 1.25 page 27), le facteur C_L (capacité équivalente d'un fil du bus) intervient pour une part importante sur la consommation. Cette capacité est composée de quatre capacités dont les plus primordiales (et surtout celles sur lesquelles une optimisation peut être effectuée) sont celle due à la capacité de la face inférieure du fil par rapport au substrat ainsi que celle due à la capacité de *crosstalk*.
La plupart des techniques développées n'ont pour but que de réduire la capacité de *crosstalk*.

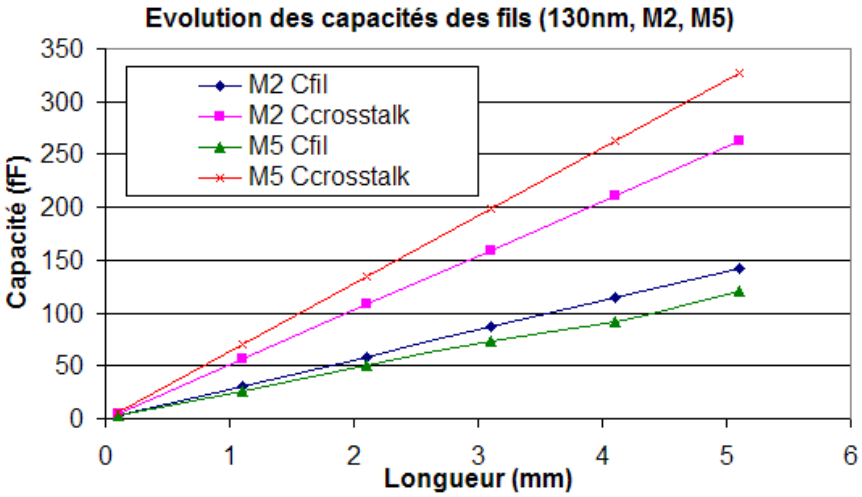


FIG. 3.1 – Evolution des capacités du fil en fonction de la longueur.

En effet comme le montre la figure 3.1 ainsi que le tableau 1.3 page 23 sur le rapport d'aspect, la capacité de *crosstalk* est toujours prépondérante par rapport à la capacité de la face inférieure du fil par rapport au substrat. Ceci s'explique par le fait que dans les technologies actuelles les interconnexions sont plus épaisses que larges.
De plus, pour les couches de métal réservées aux bus, l'espacement entre les fils (espacement qui intervient dans la capacité de *crosstalk*) est bien inférieur à la hauteur de ce fil par rapport au substrat (hauteur qui intervient dans la capacité du fil par rapport au substrat). Ceci souligne de nouveau le fait que la capacité de *crosstalk* est supérieure à celle du fil par rapport au substrat.
La capacité de *crosstalk* existe lorsque l'on se trouve dans quatre des cinq configurations du tableau 1.4 (i.e. $1+r$, $1+2r$, $1+3r$ et $1+4r$). Sur la figure 3.1 la capacité de *crosstalk* est représentée pour le cas $1+r$. Il est aisé d'imaginer l'importance du gap entre les capacités si l'on se trouve dans une transition pire cas (i.e. $1+4r$) où la capacité de *crosstalk* vue par le fil victime est multipliée alors par 4.
L'objectif de ces techniques est d'améliorer les performances en optimisant un ou plusieurs des paramètres suivants.

- Elles essaient d'éliminer les transitions pire cas du tableau 1.4 (i.e. les transitions de type $1+3r$ et $1+4r$).
- Elles essaient de réduire l'activité des données α, paramètre qui intervient dans l'équation

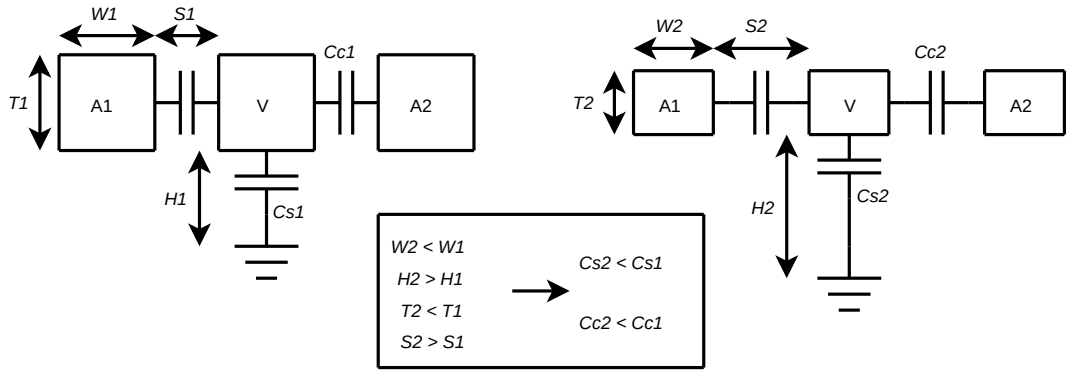


FIG. 3.2 – Impact sur les capacités dû aux modifications des dimensions des fils.

de la puissance dynamique 1.25 ;

– Elles essaient de modifier la tension d'alimentation (V_{dd}), paramètre qui intervient au carré dans l'équation de la puissance dynamique 1.25.

Un état de l'art (non exhaustif) des principales techniques d'optimisation va maintenant être proposé. Ces techniques sont classées en fonction de leur niveau d'abstraction.

3.1.1 Au niveau technologique

Modification des dimensions

Puisque les capacités qui interviennent dans la modélisation du fil sont fonction des dimensions technologiques de ce fil, une première méthode consiste ici à modifier les dimensions des fils tel que le montre la figure 3.2 :

– Hauteur (H) et largeur (W) pour réduire la capacité du fil par rapport au substrat (C_s) ;

– Epaisseur (T) et espacement (S) pour réduire la capacité de *crosstalk* (C_c).

La méthode proposée dans [MMP02] consiste à modifier les espacements entre les fils de manière non uniforme entre eux ; c'est à dire qu'entre un fil i et son voisin $i+1$ l'espacement sera de s_1 et que entre le fil $i+1$ et son voisin $i+2$, l'espacement sera de s_2 et ainsi de suite.
Les résultats obtenus montrent une diminution jusqu'à 30% de consommation énergétique sur le bus.

– L'atout de cette technique est une diminution de la consommation ainsi qu'une augmentation du débit puisque la capacité de *crosstalk* sera diminuée pour des espacements plus grands.

– Les inconvénients sont une augmentation de la surface du bus puisque l'on augmente l'espacement, ainsi qu'un problème au niveau de la fabrication puisque les dimensions modifiées ne sont, la plupart de temps, plus en concordance avec celles fournies dans le *DesignKit* du fondeur.

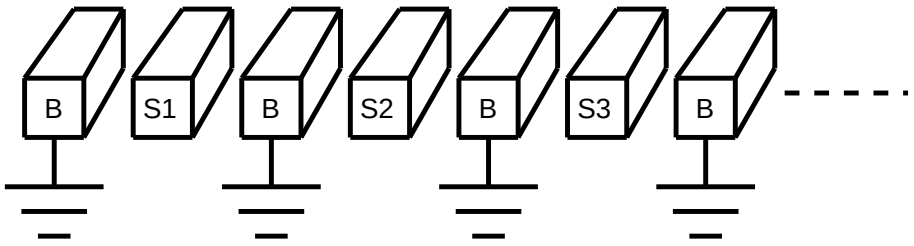


FIG. 3.3 – Illustration du blindage statique à la masse (S_i : signaux, B : fil bouclier).

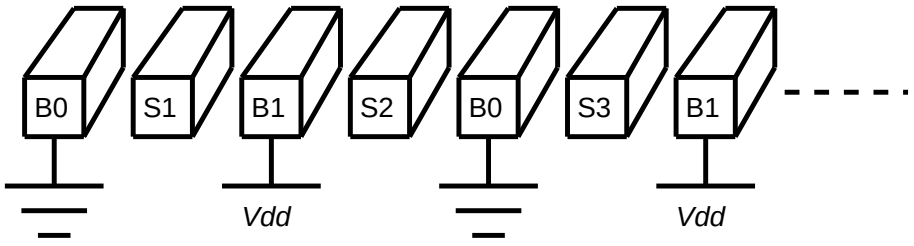


FIG. 3.4 – Illustration du blindage alterné (S_i : signaux, B_0 : fil bouclier à la masse, B_1 : fil bouclier à l'alimentation).

Blindages spatiaux

Les techniques de blindage (*shielding*) consistent à insérer des fils supplémentaires entre les fils qui véhiculent les signaux. Ces fils supplémentaires peuvent avoir des niveaux logiques fixes ou qui évoluent au cours de la transmission des données.
Dans le premier cas on parlera de blindage statique alors que dans le second de blindage dynamique.
Le premier type de blindage statique consiste à insérer entre chaque signal du bus un fil relié à la masse ou à l'alimentation tel que le montre la figure 3.3.
Ce type de blindage permet d'éliminer toutes les transitions croisées de type $1+3r$ et $1+4r$. Il en découle une forte accélération de la transmission des données sur le bus puisqu'on ne se retrouve qu'avec des transitions de type $1+2r$.
Par contre il n'existe plus de transition de type 1 ou $1+r$. En effet chaque fil victime lorsqu'il effectue une transition se voit entouré de fils dont le niveau ne change pas. Ceci a pour effet de reporter des transitions moins consommatrices dans une catégorie de transitions plus consommatrices d'un point de vue énergie. L'activité des données reste quant à elle inchangée.
Une évolution de cette technique peut être trouvée dans [KBSV78] où la technique de blindage consiste en une alternance de blindage à la masse et à l'alimentation (V_{dd}) tel que l'illustre la figure 3.4.
Les performances en termes de vitesse et de consommation sont les mêmes que pour la technique précédente car le principe est le même. L'avantage de cette technique, outre le blindage, est de permettre de distribuer le réseau d'alimentation à travers la puce selon le pattern suivant $GND, S_1, V_{dd}, S_2, GND \ldots$.
Dans [TDZ01], la technique de blindage retenue est que le fil bouclier a le niveau logique du ET logique de ses deux voisins tel que l'illustre la figure 3.5.
Puisque ici, le niveau du fil bouclier évolue à chaque transmission de donnée, le blindage est dit

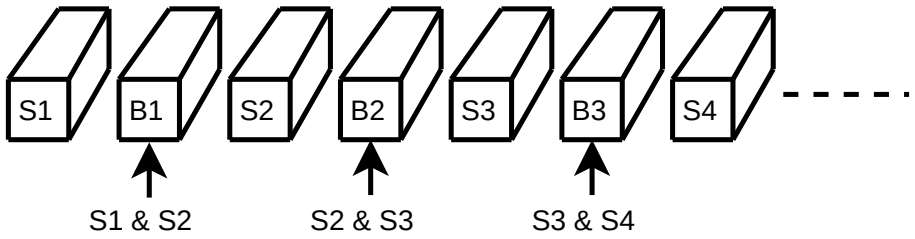


FIG. 3.5 – Illustration du blindage par ET logique ($B_i = S_i \& S_{i+1}$).

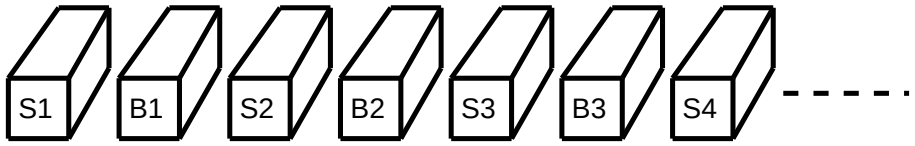


FIG. 3.6 – Illustration du blindage par duplication ($B_i = S_i$).

dynamique.

Une autre méthode très simple d'application du blindage consiste à dupliquer chaque signal en transmettant sur le fil adjacent le même signal tel que l'illustre la figure 3.6. L'accélération apportée par cette technique est supérieure à celle présentée dans [KBSV78], car le cas où les deux agresseurs sont stables est en plus éliminé.

Le principal atout des techniques de blindage est qu'elles permettent d'augmenter considérablement la vitesse de transmission des données sur le bus puisque les pires cas du tableau 1.4 sont éliminés. En contrepartie elles ne sont pas efficaces en terme de surface puisque le nombre de fils nécessaires est doublé et elles ont, de plus, un impact énergétique assez limité.

3.1.2 Au niveau circuit

Décalage temporel des signaux (*signal skewing*)

La solution présentée dans [HY00] consiste à décaler intentionnellement les signaux du bus pour éviter d'avoir des transitions simultanées sur deux fils voisins.

Les fils pairs et impairs sont déphasés temporellement les uns par rapports aux autres. Ainsi, un fil (qu'il soit pair ou impair) effectuera sa transition lorsque ses deux voisins seront stables comme le montre la figure 3.7.

De cette manière, les transitions pire cas seront des transitions de type $1 + 2r$.

L'accélération apportée par cette technique est toutefois limitée par le fait que la transmission d'une donnée est ralentie. En effet, un temps d'attente est rajouté entre la transmission des bits pairs et impairs. Les résultats de simulation montrent que l'accélération peut être comprise entre 5 et 20%.

Cette technique n'est basée que sur des simulations et a besoin d'une conception compliquée au niveau de l'horloge pour les émetteurs et les récepteurs ; de plus aucune implantation n'est proposée.

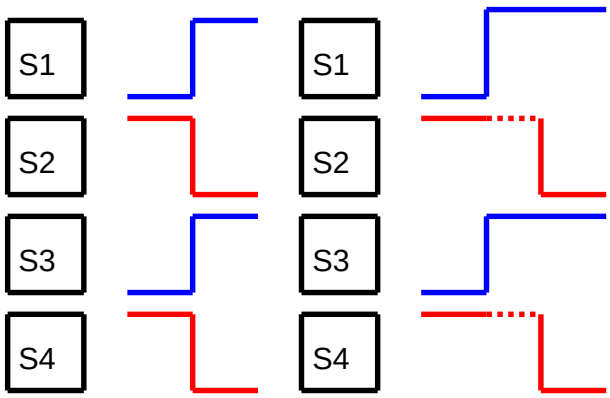


FIG. 3.7 – Illustration du décalage temporel (*signal skewing*).

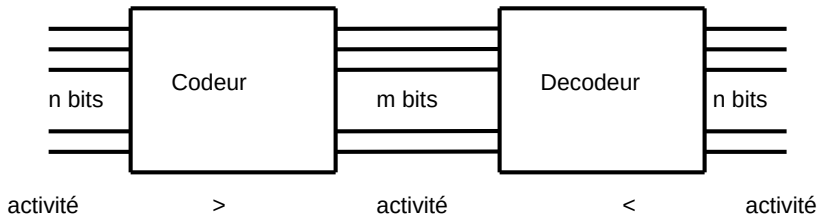


FIG. 3.8 – Principe du codage des données.

Variation de la tension d'alimentation des buffers.

Il est proposé dans [SPJ03] d'utiliser plusieurs valeurs de tension d'alimentation pour les buffers. L'utilisation de cette technique a pour but de limiter les excursions de tension sur les lignes.
Le principe de la méthode est de faire de l'adaptation dynamique de la tension d'alimentation (*Dynamic Voltage Scaling : DVS*) de ces répéteurs en fonction de la fréquence de fonctionnement qui leur est imposée.
Les résultats de simulation montrent ici une réduction de 4.6x sur la puissance dynamique (en moyenne) accompagnée d'une diminution de 15.2% sur la latence.
Par contre, cette technique nécessite l'ajout de plusieurs blocs analogiques de contrôle ayant pour but la gestion de l'adaptation de tension.

3.1.3 Au niveau architectural

C'est au niveau architectural que l'on rencontre le plus grand nombre de techniques d'optimisation des performances.
Elles consistent toutes en un codage des données tel que le montre la figure 3.8. Le but du codage est de transmettre l'information originale codée sur n bits en une information codée sur m bits avec $n \leqslant m$ tel que l'activité α des données codées soit inférieure à celle des données non codées.
Dans un premier temps nous allons introduire les techniques propres au bus d'adresses puis nous verrons celles propres au bus de données.

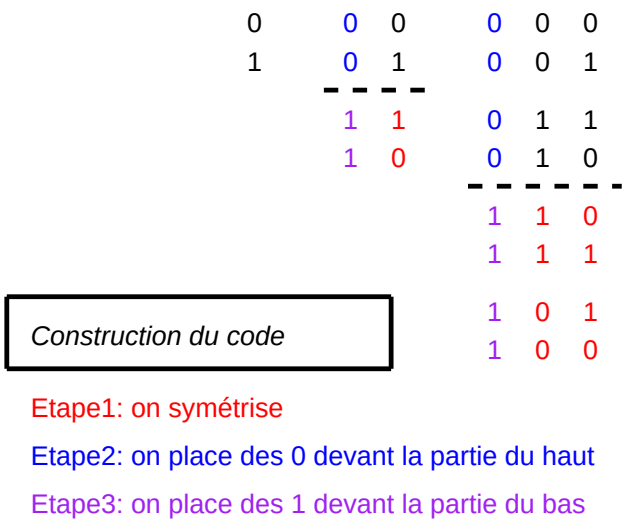


FIG. 3.9 – Construction du codage de *Gray*.

Cycle	Adresse	Transitions	Gray(Adresse)	Transitions
1	0000	0	0000	0
2	0001	1	0001	1
3	0010	2	0011	1
4	0011	1	0010	1
5	0100	3	0110	1
6	0101	1	0111	1
7	0110	2	0101	1
8	0111	1	0100	1
9	1000	4	1100	1
10	1001	1	1101	1

TAB. 3.1 – Différence entre le nombre de transitions en binaire pur et en code de *Gray* pour des adresses consécutives.

Codage de *Gray*

L'idée du codage présenté dans [STD94], [SD95] est de faire en sorte que l'on ait une seule transition sur le bus à chaque fois que l'on accède à une adresse consécutive de celle accédée au cycle précédent. Ce codage est appelé codage de *Gray* ou code binaire réfléchi.
La construction du code est illustrée sur la figure 3.9.
Le tableau 3.1 montre la différence entre le nombre de transitions en binaire pur et en code de *Gray*. Lorsque les adresses accédées sont consécutives, il n'y a bien qu'une seule transition avec codage.
Au niveau du codeur l'équation qu'il faut appliquer pour déterminer la valeur du bit i du bus est la suivante :

$$G_n = B_n \oplus B_{n+1} \tag{3.1}$$

– G_n représente la valeur du bit n codé ;

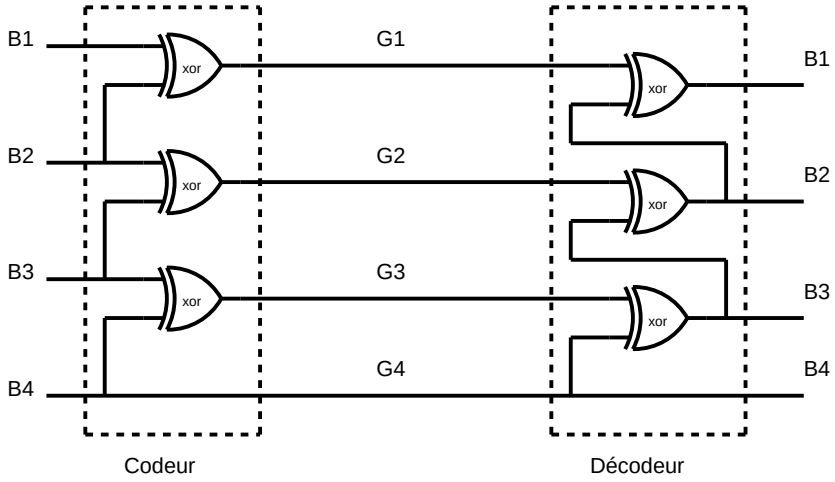


FIG. 3.10 – Architecture des codeur et décodeur pour le code de *Gray*. Exemple d'un bus de 4 bits.

- B_n représente la valeur du bit n non codé ;
- B_{n+1} représente la valeur du bit $n+1$ non codé.

Au niveau du décodeur l'équation qu'il faut appliquer pour retrouver le codage en binaire pur est la suivante :

$$B_n = G_n \oplus B_{n+1} \tag{3.2}$$

A partir de ces formules, il est aisé de construire l'architecture du codeur et du décodeur tel que le montre la figure 3.10.

Les expérimentations effectuées dans [SD95] montrent une réduction de l'activité de 33% et une réduction de l'énergie consommée sur le bus de 77%.

Par contre pour des bus larges (quand n est grand), le décodeur possède un long chemin critique puisque les portes ou-exclusives sont cascadées des *MSB* vers les *LSB*.

Code *T0*

Dans [BMM+97], l'idée proposée est de rajouter un fil noté *INC* que l'on positionne à un niveau logique défini lorsque les adresses accédées sont consécutives. Pour cela, la valeur de l'adresse au cycle d'horloge $t-1$ est stockée, une incrémentation de 1 est effectuée puis cette valeur est comparée à celle arrivant au cycle d'horloge t. Si c'est deux valeurs sont identiques alors l'état du bus ne change pas et le fil supplémentaire *INC* est positionné à un niveau logique défini. Dans le cas contraire, la valeur de l'adresse est envoyée sur le bus. Au niveau du décodeur une sélection est effectuée en fonction de l'état du fil *INC* entre la valeur sur le bus ou la valeur passée incrémentée de 1.

Cette technique réduit l'activité à 0 lorsque les adresses accédées sont consécutives, ce qui permet de réduire fortement la consommation sur le bus.

Une évolution de cette technique est proposée dans [FPSS00] où il est possible de définir plusieurs pas d'incrémentation (*+step*) pour des accès consécutifs.

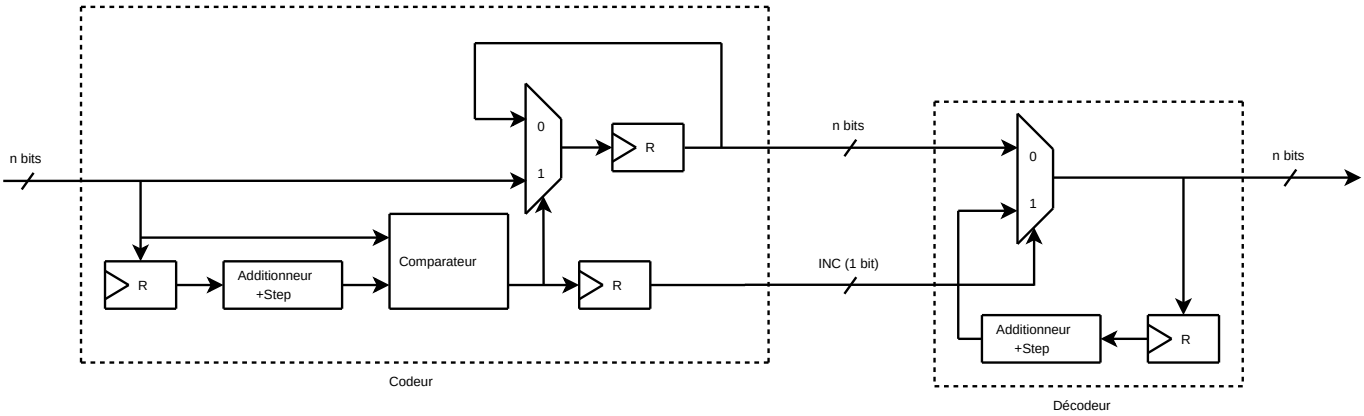


FIG. 3.11 – Architecture des codeur et décodeur pour le code *T0*.

Une seconde évolution du Code *T0* est introduite dans [AFP01] où il est proposé en rajoutant de la logique supplémentaire de supprimer la ligne de contrôle *INC*. Malheureusement la conception des codeur et décodeur est assez complexe (banc de registres, additionneurs, multiplexeurs...) tel que le montre la figure 3.11. De plus, le surcoût en consommation des codecs (non évaluée dans [BMM+97]) est supérieur à la réduction de consommation sur le bus apportée par le codage pour des longueurs d'interconnexions utilisées dans les SOC actuels.

3.1.4 Au niveau architectural pour le bus de données

Bus Invert / Partial Bus Invert

L'idée du codage présenté dans [SB95] et illustré sur la figure 3.12 est de comparer le nombre de bits changeant entre la donnée $n-1$ au cycle d'horloge $t-1$ et la donnée n au cycle t. Si cette différence (que l'on nomme distance de *Hamming*) est supérieure à la moitié de la largeur du bus alors la donnée envoyée au cycle t sera complémentée et on enverra $\overline{n}$. Il faudra également signaler au décodeur par l'intermédiaire d'une ligne supplémentaire notée INV positionnée à un niveau logique défini s'il doit faire le complément sur la donnée qu'il reçoit.
Cette technique appelée le *Bus Invert* est efficace pour de larges bus mais ne doit être appliquée que sur les bits ayant une forte activité. En effet si l'on applique le *Bus Invert* sur tout le bus, son utilisation risque d'être moins fréquente car les *MSB* sont souvent corrélés entre eux. La probabilité qu'il y ait 50% de transitions sur le bus sera plus faible que si l'on applique la technique sur les bits ayant la plus forte activité à savoir les *LSB*.
C'est pourquoi [SCC98] propose d'appliquer la technique du *Bus Invert* à la seule partie du bus qui possède la plus forte activité. On appelle ce codage le *Partial Bus Invert*.
L'inconvénient du *Partial Bus Invert* est qu'il faut connaitre à l'avance où l'activité la plus forte sur le bus aura lieu. Il faut donc connaitre les données qui vont circuler sur le bus.
Au niveau du codeur, l'équation qu'il faut appliquer est la suivante :

$$X_n = B_n \oplus INV \tag{3.3}$$

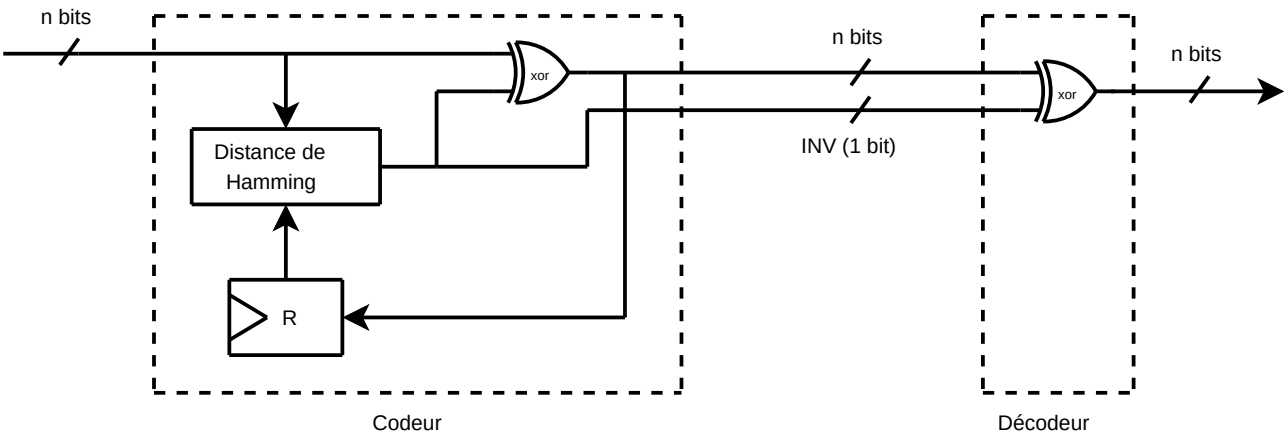


FIG. 3.12 – Architecture des codeur et décodeur pour le code *Bus Invert*.

- Si la distance de Hamming est positive alors $INV = 1$ et $X_n = \overline{B}_n$;
- Sinon $INV = 0$ et $X_n = B_n$.

Au niveau du décodeur l'équation qu'il faut appliquer pour retrouver la donnée originale est la suivante :

$$B_n = X_n \oplus INV \tag{3.4}$$

- Si $INV = 1$ alors $B_n = \overline{X}_n$;
- Sinon $INV = 0$ et $B_n = X_n$.

Dictionnaire de codes (*Code Book*)

Le principe du codage présenté dans [KIA99][LHLW03] appelé *Code Book* vise à stocker i anciennes valeurs transmises sur le bus et à transmettre au cycle courant la valeur qui a le moins de différences avec celles transmises aux i cycles précédents.
Il est alors également nécessaire de transmettre sur plusieurs fils supplémentaires (2^i fils) le code de la valeur envoyée afin de signaler au décodeur avec quelle donnée il doit faire la reconversion.
La figure 3.13 illustre l'architecture de la technique du *Code Book* pour $i = 3$.
Les résultats expérimentaux présentés, montrent une diminution de la consommation pour des bus extrèmement longs (plus de 7.5cm), longueur inatteignable dans un *SoC*.

Code 0 (Blindage temporel)

Le blindage temporel présenté dans [PPS06] consiste, quant à lui, à faire retomber le bus à 0 entre chaque donnée transmise tel que l'illustre la figure 3.14. De cette manière, il est impossible qu'il se produise de transitions croisées de type $1 + 3r$ ou $1 + 4r$. Cette technique permet de réduire la capacité présentée par le bus puisque les transitions pire cas seront des transitions de type $1 + 2r$. En revanche, les résultats présentés par les auteurs montrent des performances en termes de consommation négatives (augmentation). En effet, il faut transmettre deux fois plus de données que dans un cas sans codage.
Le fait de faire retomber le bus à 0 entre chaque donnée transmise va introduire des transitions

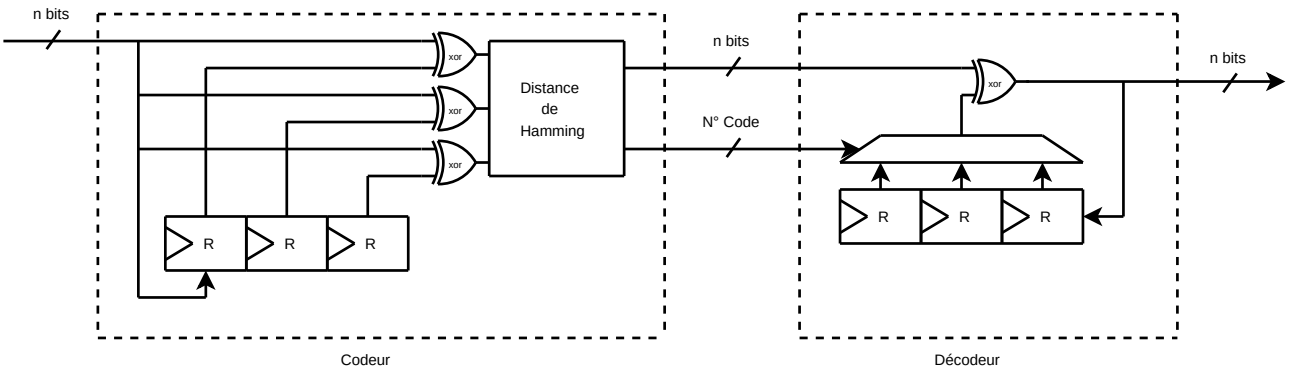


FIG. 3.13 – Architecture des codeur et décodeur pour le *Code Book*.

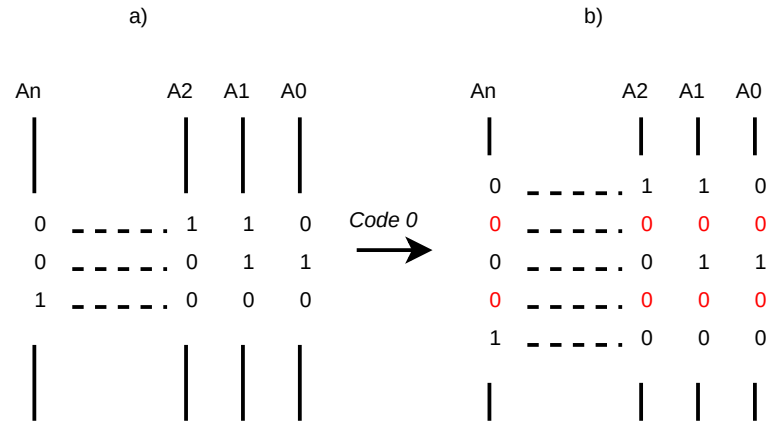


FIG. 3.14 – Construction du *Code0* ; a) sans codage, b) avec codage.

	Code 1	Code 2	
Bloc original	Codage	Codage1	Codage2
00	0000	0001	0000
01	0001	0011	1000
10	0011	0111	1100
11	0111	1111	1110

TAB. 3.2 – Correspondance entre les blocs originaux et les blocs codés.

non voulues (augmentation du taux d'activité) et de ce fait la consommation énergétique se verra également augmentée. De plus, le fonctionnement des codecs requiert une fréquence doublée par rapport au cas sans codage car il faut transmettre deux données pour une donnée utile.

Code 1

L'inconvénient du *Code 0* est que l'activité des données codées est augmentée. Les auteurs de [PPS06] proposent alors un autre codage appelé le *Code 1* qui propose également un codage temporel des données tout en essayant de ne pas (ou moins) augmenter l'activité des données.
Le principe du *Code 1* est de coder un paquet de deux bits arrivant sur un fil en un paquet de quatre bits tel que le montre le tableau 3.2. L'intérêt de ce code est qu'il permet également d'éviter les transitions de type $1 + 3r$ et $1 + 4r$. L'activité des données codées par la technique du *Code 1* est encore légèrement supérieure à l'activité des données originales, mais les résultats montrent un gain en consommation énergétique sur le bus atteignant 6.7% pour un fil de 1mm.

Code 2

Afin d'éviter l'augmentation d'activité lors du codage des données, les auteurs proposent une évolution du *Code 1*. Pour une même séquence de deux bits, le *Code 2* propose deux codages de quatre bits. Lors de la transmission de deux séquences de deux bits consécutives, la première séquence de codage puis la seconde sera transmise alternativement et ainsi de suite.
L'activité des données codées est alors la même que celles des données originales avec comme transition pire cas des transitions de type $1+2r$. Le gain en consommation sur le bus est supérieur à celui des deux codages précédents (gain jusqu'à 18.7% en consommation sur le bus).
Les auteurs de ces codages temporels ne proposent malheureusement pas d'évaluation du coût en consommation des codecs.

Codes mixés

Pour des bus présentant une communication particulière, il est possible d'utiliser des combinaisons des codages présentés précédemment. Par exemple, pour des bus partagés (bus d'adresses et de données multiplexés temporellement), les travaux présentés dans [BMM+98] proposent d'utiliser le *code T0* lors de la transmission d'une adresse au cycle d'horloge i, puis successivement d'utiliser le *Bus Invert* lors de la transmission d'une donnée au cycle d'horloge $i + 1$ et ainsi de suite.

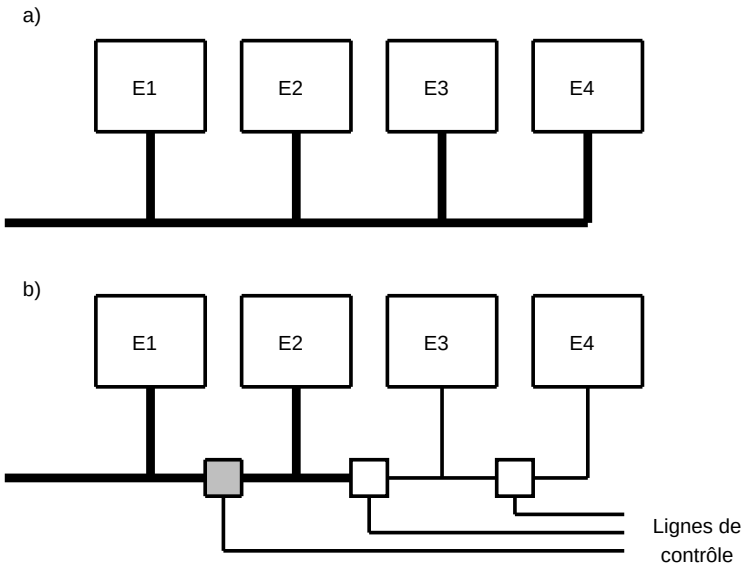


FIG. 3.15 – Illustration de la segmentation. a) E1 communique avec E2 sans segmentation. b) E1 communique avec E2 avec segmentation.

3.1.5 Au niveau système

Segmentation du bus

L'idée proposée dans [CJW+99] est de modifier la topologie d'un bus partagé entre plusieurs entités. Le problème souligné est que si deux entités, très proches spatialement sur le bus, doivent communiquer intensément, alors il faudra quand même propager les signaux sur toute la longueur du bus et donc par la même occasion faire des charges et décharges complètes du bus.
Le bus est alors ici segmenté (tel que l'illustre la figure 3.15), chaque segment étant isolé de son voisin par des blocs *pass transistor*. On placera donc sur un même segment les entités qui ont souvent besoin de communiquer entre elles.
Les résultats expérimentaux montrent une diminution jusqu'à 70% de la puissance consommée ainsi qu'une diminution du délai jusqu'à 30% sur le bus.

Mécanismes d'échange des données (*Equal split, Don't care padding*)

Les travaux de [GV98] proposent des mécanismes d'échange des données afin de limiter le nombre de transitions sur le bus.

Equal Split Dans un système, les différentes entités ne communiquent pas forcément sur le même nombre de bits. Le premier mécanisme d'échange de données concerne un envoi de données dont la taille en nombre de bits dépasse celle du bus. Une première possibilité de transmission est de couper la donnée en deux ; au premier cycle tout le bus est utilisé et au second cycle seulement une partie du bus est utilisée, le reste des lignes est forcé à 0. Le principe du mécanisme de l'*equal split* est de couper la donnée à envoyer en deux données de taille identique et de les envoyer sur deux cycles. Pour illustrer le fonctionnement de l'equal split, imaginons une donnée de 8 bits à envoyer sur un bus de 5 bits :

- Sans *equal split*, le premier transfert envoie les 5 premiers bits et le second les 3 derniers avec deux lignes forcées à 0. Soit un total de 10 bits transmis et potentiellement 5 transitions par cycle.
- Avec *equal split*, le premier transfert envoie les 4 premiers bits et le second les 4 derniers avec une ligne qui ne commutera pas. Soit un total de 8 bits transmis et potentiellement 4 transitions par cycle.

Don't care padding Le second mécanisme d'échange de données concerne un envoi de deux données dont la taille en nombre de bits est inférieure à celle du bus pour la seconde donnée. Une première possibilité de transmission est au premier cycle d'envoyer la première donnée sur tout le bus et au second cycle d'utiliser seulement une partie du bus en forçant le reste des lignes à 0. Le principe du mécanisme du *Don't care padding* est le suivant : lors de la transmission de la seconde donnée, les lignes du bus non utiles sont laissées dans l'état précédent. Pour illustrer le fonctionnement du *Don't care padding*, imaginons une première donnée de 8 bits suivie d'une de 5 bits à envoyer sur un bus de 8 bits :

- Sans *Don't care padding*, le premier transfert envoie les 8 premiers bits et le second les 5 derniers avec 3 lignes forcées à 0. Soit un total de 16 bits transmis et potentiellement 8 transitions par cycle.
- Avec *Don't care padding*, le premier transfert envoie les 8 premiers bits et le second les 5 derniers avec 3 lignes dont on ne change pas l'état. Soit un total de 13 bits transmis et potentiellement 6.5 transitions par cycle en moyenne.

Les résultats expérimentaux en combinant ces techniques annoncent une réduction de consommation de 22 à 47% pour des données aléatoires dans le cadre du sytème utilisé.

3.1.6 Bilan : Impact des techniques d'optimisation et analyse par Interconnect Explorer

Dans cette section, un récapitulatif de l'effet des techniques d'optimisation va être présenté, tant d'un point de vue consommation qu'en terme de délai. Cette étude porte uniquement sur les techniques pour le bus de données.

Le tableau 3.3 propose un bilan des avantages et inconvénients des techniques d'optimisation présentées dans cette section. Les variations de la consommation d'énergie et/ou de puissance, de l'activité et du temps de propagation sont quantifiées numériquement quand cela est présenté dans les travaux. Si seule une tendance est annoncée, cela est représenté par une flèche ; si rien n'est indiqué un point d'interrogation précise que l'information n'est pas disponible. Enfin la dernière colonne du tableau propose quelques commentaires sur les avantages et inconvénients de ces techniques.

Afin d'évaluer quelles sont les techniques d'optimisation les plus performantes, *Interconnect Explorer* a été utilisé. Les résultats ont été obtenus en utilisant des stimuli de type image, musique et parole dans une technologie 65nm pour deux couches de métal (métal 2 et 4) et pour une longueur de bus de 1mm. Le tableau 3.4 montre les résultats d'estimation sur la variation de l'activité, de la capacité pire cas, de la surface du bus (sans la surface due aux codecs), du temps

Technique	Puissance (P) / Energie (E)	Temps de propagation	Activité	Commentaires
Wire spacing [MMP02]	jusqu'à 40% bus E ↓	?	?	Pas de logique supplémentaire Surface bus ↑
Shielding V_{dd}/GND [KBSV78]	?	?	?	Pas de logique supplémentaire Surface bus ↑ Distribution lignes d'alimentation
Shielding AND [TDZ01]	jusqu'à 50% bus E ↓	↓	↑	Surface bus ↑
DVS [SPJ03]	4.6x P ↓ en moyenne	↑	?	Logique supplémentaire complexe Latence bus ↑
Signal skewing [HY00]	?	5 à 20% ↓	?	Pas d'implantation proposée
Code de Gray [STD94], [SD95]	jusqu'à 77% bus P ↓	?	33% ↓ bus @ instruction 12% ↓ bus @ donnée	Latence ↑ avec la largeur du bus Surface bus ↑
Code T0 [BMM+97]	$P_{dyn} = 0$ si @ sequentielles	?	↓	Logique supplémentaire complexe Ligne en plus
Bus Invert [SB95]	jusqu'à 50% I/O P_{crte} ↓ jusqu'à 25% I/O$P_{moyenne}$ ↓	?	↓	Logique supplémentaire complexe Ligne en plus
Partial Bus Invert [SCC98]	?	?	62.5% ↓	Idem Bus invert
Code Book [KIA99]	?	?	25 à 50% ↓	Logique supplémentaire complexe Ligne(s) en plus
Code 0 [PPS06]	jusqu'à 21.5% bus E ↑	↓	↑	Besoin d'une fréquence double Bande passante ↑ 31.8% Pas d'implantation proposée
Code 1 [PPS06]	jusqu'à 10.7% bus E ↓	↓	↑	
Code 2 [PPS06]	jusqu'à 18.7% bus E ↓	↓	–	
Bus segmenting [CJW+99]	jusqu'à 70% P ↓	jusqu'à 30% ↓	?	Ligne(s) en plus
Equal split	22 à 47% P ↓	?	?	Mise en oeuvre simple
Don't care padding [GV98]				

TAB. 3.3 – Bilan des effets des techniques d'optimisation avec les données issues de la littérature.

de propagation et de l'énergie consommée sur le bus. Dans ce tableau la technique du *Partial Bus Invert* est le *Bus Invert* appliqué aux trois derniers *LSB*. Les résultats montrent que toutes les techniques offrent une réduction du temps de propagation sur le bus. La meilleure réduction de consommation est obtenue pour la technique du *Partial Bus Invert* (11.4% de réduction de consommation sur le bus). Les résultats, en termes de réduction de la consommation, obtenus par les autres techniques, montrent qu'elles ne sont maintenant plus efficaces pour les technologies et longueurs de bus actuels dans les SoC.

3.2 De nouvelles pistes d'optimisation

Comme nous venons de le voir, dans la littérature beaucoup de techniques visent à réduire l'effet du *crosstalk* ayant le but d'accélérer la propagation des informations et également de réduire la consommation des interconnexions.
Ces techniques interviennent à plusieurs niveaux d'abstraction. Elles se basent sur le tableau 1.4 en considérant que le classement des transitions en temporel sera le même que celui d'un point de vue consommation.
Afin de confirmer la pertinence de ces méthodes, nous nous sommes attachés à vérifier si les transitions pire cas en temporel sont bien les pires cas en consommation. Les expérimentations effectuées montrent que cela n'est pas toujours le cas.

3.2.1 Validité du tableau des transitions

Dans l'état de l'art sur les techniques d'optimisation de la consommation pour les interconnexions, la plupart des techniques proposées visent à éliminer les classes de transition les plus pénalisantes du tableau 1.4 (i.e. $1+3r$ et $1+4r$). Ces techniques prennent comme point de départ que les transitions les plus pénalisantes du tableau 1.4 sont également les plus pénalisantes en termes de consommation. Nous avons donc vérifié à l'aide de notre outil si tel était bien le cas afin de proposer une éventuelle nouvelle classification.
Les expérimentations présentées ont été réalisées en utilisant l'outil sur différentes couches de métal réservées aux bus pour une longueur de 1mm dans une technologie 65nm (notons que les résultats sont exactement similaires pour les autres technologies).
Le tableau 3.5 montre dans sa partie gauche le temps de propagation et dans sa partie droite la consommation pour les différents types de transitions. Dans un premier temps, les résultats du tableau 3.5 montrent que la classification temporelle en fonction de l'importance de la capacité présentée par le fil victime est la même que celle présentée dans le tableau 1.4. Dans un second temps, il est important de remarquer que la classification des transitions d'un point de vue consommation n'est pas similaire à celle du point de vue temporel.
La classification en consommation du tableau 3.5 peut se diviser en deux parties :

- dans la partie haute du tableau, les transitions sont exclusivement montantes et sont classées de celle présentant la plus faible capacité à celle présentant la plus forte ;
- dans la partie basse du tableau, les transitions sont exclusivement descendantes et sont

Techniques pour le bus de donnée	Variation (%) Activité	Capacité pire cas C_L	Variation (%) Surface du bus	Variation (%) Temps de propagation		Variation (%) Consommation E bus	
				Metal 2	Metal 4	Metal 2	Metal 4
Shielding GND	0	$C_s + 2C_c$	↑ 107.8	↓ 42.3	↓ 42.6	↓ 5.2	↓ 6.2
Shielding V_{dd}/GND	0	$C_s + 2C_c$	↑ 107.8	↓ 41.5	↓ 41.8	↓ 5.2	↓ 6.2
Shielding AND	↓ 18.6	$C_s + 2C_c$	↑ 107.8	↓ 42.3	↓ 42.6	↓ 1.9	↓ 2.6
Duplication	0	$C_s + 2C_c$	↑ 107.8	↓ 42.9	↓ 43.2	↑ 30.7	↑ 30.2
Bus Invert	↓ 11	$C_s + 4C_c$	↑ 13.5	0	0	↓ 7.9	↓ 7.8
Partial Bus Invert	↓ 19.5	$C_s + 4C_c$	↑ 13.5	0	0	↓ 11.4	↓ 11.4
Code 0	↑ 40	$C_s + 2C_c$	0	↓ 42.3	↓ 42.6	↑ 76.2	↑ 75.8
Code 1	↓ 5.1	$C_s + 2C_c$	0	↓ 40.8	↓ 41.0	↑ 42.2	↑ 40.5
Code 2	↑ 1	$C_s + 2C_c$	0	↓ 40.8	↓ 41.0	↑ 7.6	↑ 6.4

TAB. 3.4 – Bilan des effets des techniques d'optimisation analysées par *Interconnect Explorer*.

Classification Temporelle		
$(-,-,-)$	0	0 ps
$(\uparrow,\uparrow,\uparrow)$	C_s	49 ps
$(\downarrow,\downarrow,\downarrow)$	C_s	49 ps
$(-,\uparrow,\uparrow)$	$C_s + C_c$	67 ps
$(-,\downarrow,\downarrow)$	$C_s + C_c$	67 ps
$(\uparrow,\uparrow,\downarrow)$	$C_s + 2C_c$	99 ps
$(-,\uparrow,-)$	$C_s + 2C_c$	99 ps
$(-,\downarrow,-)$	$C_s + 2C_c$	99 ps
$(\downarrow,\downarrow,\uparrow)$	$C_s + 2C_c$	99 ps
$(-,\uparrow,\downarrow)$	$C_s + 3C_c$	139 ps
$(-,\downarrow,\uparrow)$	$C_s + 3C_c$	139 ps
$(\downarrow,\uparrow,\downarrow)$	$C_s + 4C_c$	173 ps
$(\uparrow,\downarrow,\uparrow)$	$C_s + 4C_c$	173 ps

Classification Energétique		
$(-,-,-)$	0	0.21 fJ
$(\uparrow,\uparrow,\uparrow)$	C_s	13.29 fJ
$(-,\uparrow,\uparrow)$	$C_s + C_c$	13.43 fJ
$(-,\uparrow,-)$	$C_s + 2C_c$	13.45 fJ
$(\uparrow,\uparrow,\downarrow)$	$C_s + 2C_c$	13.89 fJ
$(-,\uparrow,\downarrow)$	$C_s + 3C_c$	14.10 fJ
$(\downarrow,\uparrow,\downarrow)$	$C_s + 4C_c$	14.86 fJ
$(\downarrow,\downarrow,\downarrow)$	C_s	33.77 fJ
$(-,\downarrow,\downarrow)$	$C_s + C_c$	92.00 fJ
$(-,\downarrow,-)$	$C_s + 2C_c$	150.35 fJ
$(\downarrow,\downarrow,\uparrow)$	$C_s + 2C_c$	150.73 fJ
$(-,\downarrow,\uparrow)$	$C_s + 3C_c$	207.76 fJ
$(\uparrow,\downarrow,\uparrow)$	$C_s + 4C_c$	265.07 fJ

TAB. 3.5 – Classification des transitions d'un point de vue délai et consommation énergétique.

classées de la même manière.

Il est important de noter que la consommation des transitions montantes est similaire, alors que pour les transitions descendantes, la consommation augmente avec la taille de la capacité.

Afin de comprendre pourquoi les transitions descendantes consomment plus que les montantes, il est nécessaire de savoir quand le courant consommé provient de l'alimentation.

Deux cas peuvent être considérés, les lignes d'interconnexions peuvent être simplement bufferisées (un buffer en entrée et un en sortie de la ligne) ou complètement bufferisées tel que présenté dans le chapitre précédent. En fonction du type de transition (montante ou descendante), la capacité présentée par la ligne (ou le segment dans le cas d'une bufferisation complète) est chargée ou non comme l'illustre la figure 3.16.

- Pour la bufferisation simple (figure 3.16a)), lors d'une transition montante, c'est le transistor NMOS qui est actif et donc la capacité présentée par la ligne n'est pas chargée par l'alimentation. Lors d'une transition descendante, c'est le transistor PMOS qui est actif et donc la capacité présentée par la ligne est chargée par le courant extrait de l'alimentation ;
- Pour la bufferisation complète (figure 3.16b)), le nombre de buffers insérés le long des lignes doit être pair afin que le signal à la sortie de la ligne soit le même qu'à l'entrée. Par conséquent, il y aura toujours un segment de plus à charger lors d'une transition descendante comparé à une montante ; ce qui explique que les transitions descendantes soient ici également plus pénalisantes en terme de consommation.

Les expérimentations permettent de conclure que la classification des transitions les plus pénalisante en terme de temps de propagation n'est pas la même que celle d'un point de vue consommation.

Dans le cas de transitions montantes, la consommation varie seulement de 5.6% autour de la valeur moyenne, elles peuvent donc être toutes classées dans la même catégorie. En fait, la consommation des transitions montantes est due au chemin de court-circuit entre l'alimentation et la masse durant la transition. Durant une transition descendante (donc montante en sortie du premier in-

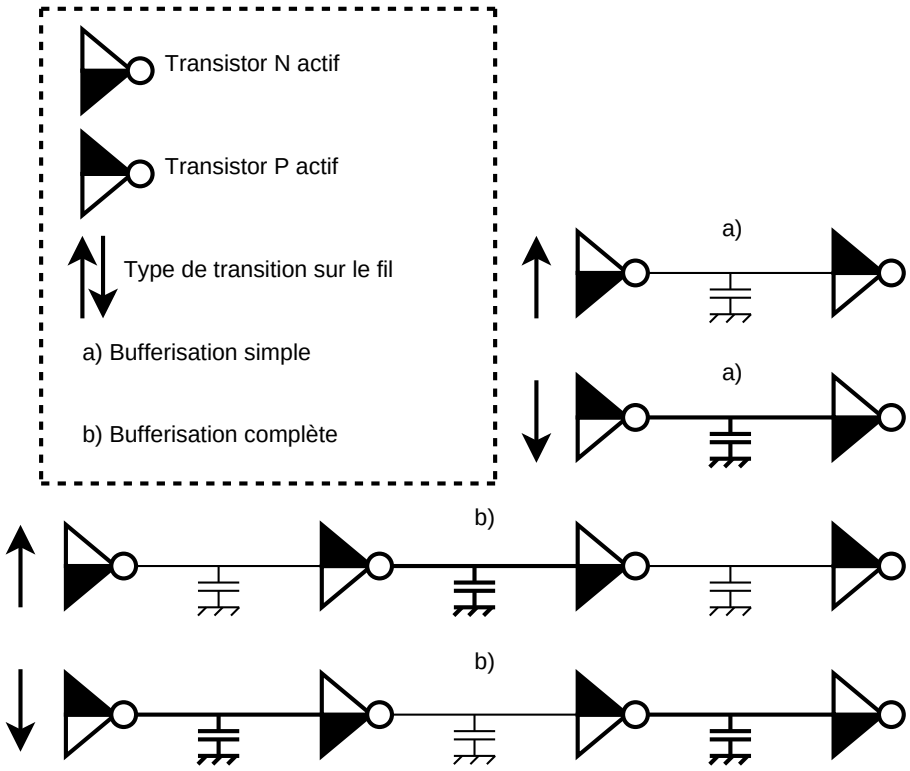


FIG. 3.16 – Ligne ou segments chargés (en gras) en fonction de la bufferisation et du type de transition.

$C_{L(i,j)}$	Transitions	Energie	j
X	$(?,\uparrow,?)$	$E_{court-circuit}$	X
C_s	$(\downarrow,\downarrow,\downarrow)$	$E_{i,j} = E_{i,0}$	0
$C_s + C_c$	$(-,\downarrow,\downarrow)$ $(\downarrow,\downarrow,-)$	$E_{i,j} = E_{i,1}$	1
$C_s + 2C_c$	$(-,\downarrow,-)$ $(\downarrow,\downarrow,\uparrow)$ $(\uparrow,\downarrow,\downarrow)$	$E_{i,j} = E_{i,2}$	2
$C_s + 3C_c$	$(-,\downarrow,\uparrow)$ $(\downarrow,\uparrow,-)$	$E_{i,j} = E_{i,3}$	3
$C_s + 4C_c$	$(\uparrow,\downarrow,\uparrow)$	$E_{i,j} = E_{i,4}$	4

TAB. 3.6 – Classification des transitions d'un point de vue de la consommation énergétique.

verseur), le courant extrait de l'alimentation pour charger la capacité présentée par le fil dépend du type de transition et donc de la capacité du fil. Cette énergie accumulée est ensuite restituée dans la masse pendant le changement d'état lors de la prochaine transition.
Un point clé pour les optimisations futures pourrait être de coder les données de telle sorte que les transitions descendantes sur le bus soient celles qui présentent la plus faible capacité de *crosstalk* (i.e. transition $(\downarrow,\downarrow,\downarrow)$) et soient donc les moins consommatrices.
Le tableau 3.6 présente la classification des transitions en consommation à laquelle nous avons abouti. Avec cette nouvelle classification, un nouveau modèle de consommation dynamique plus précis peut être établi. L'énergie consommée sur un bus N_{bit} peut être définie par la formule

suivante :

$$E_{dynamique} = \sum_{i=0}^{N_{bit}-1} \sum_{j=0}^{4} [P_{i,j} E_{i,j}] \tag{3.5}$$

- $P_{i,j}$ est la probabilité d'avoir une transition de type j sur le fil i ;
- $E_{i,j}$ est la consommation énergétique correspondante.

Notons que j varie de 0 à 4 alors que i de 0 à $N_{bit} - 1$ comme défini dans le tableau 3.6. Pour un cycle de transition complet, s'il y a une transition montante (ou descendante), il y a forcément plus tard une transition descendante (ou montante respectivement). La consommation énergétique $E_{i,j}$ peut donc alors être calculée par la formule suivante :

$$E_{i,j} = E_{court-circuit} + C_{L(i,j)} V_{dd} V_{swing} \tag{3.6}$$

- $E_{court-circuit}$ est la consommation énergétique due au chemin de court-circuit entre l'alimentation et la masse pendant le changement d'état lors de la transition ;
- $C_{L(i,j)} V_{dd} V_{swing}$ est la consommation énergétique due au chargement de la capacité présentée par le fil.
  - V_{dd} représente la tension d'alimentation ;
  - V_{swing} représente la tension d'excursion de la transition ;
  - $C_{L(i,j)}$ représente la capacité à charger pour une transition de type j sur le fil i. Cette capacité peut être calculée de la manière suivante : $C_{L(i,j)} = C_s + jC_c$ comme représenté dans le tableau 3.6 avec $j \in [0, 4]$.

La consommation de puissance dynamique s'exprime donc comme suit :

$$P_{dynamique} = E_{dynamique} F \tag{3.7}$$

où F représente la fréquence d'envoi des données sur le bus. En substituant l'équation 3.5 dans l'équation 3.7, $P_{dynamique}$ devient :

$$P_{dynamique} = \sum_{i=0}^{N_{bit}-1} \sum_{j=0}^{4} [P_{i,j} E_{i,j} F] \tag{3.8}$$

$$P_{dynamique} = \sum_{i=0}^{N_{bit}-1} \sum_{j=0}^{4} \left[P_{i,j} F \left(E_{court-circuit} + C_{L(i,j)} V_{dd} V_{swing}\right)\right] \tag{3.9}$$

Nous allons dans la partie suivante nous attacher à localiser sur les bus où se trouve la partie prépondérante de la consommation. Ceci permettra alors de porter les optimisations là où la consommation est critique.

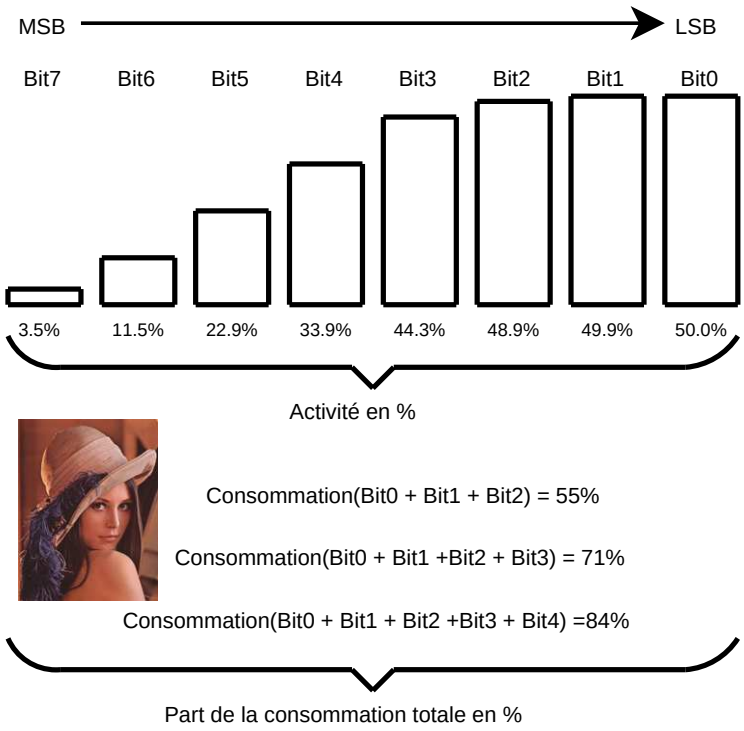


FIG. 3.17 – L'activité de chaque fil sur un bus ainsi que la part de la somme de consommation pour plusieurs bits de poids faible par rapport à la consommation totale pour un flot de données de type image et un bus de 8 bits sont représentés.

3.2.2 Où se situe la consommation ?

La consommation sur les bus est essentiellement de la consommation dynamique qui dépend entre autre de la capacité présentée par la ligne, comme il a été vu précédemment, mais également de l'activité des données sur cette ligne.

La figure 3.17 présente l'activité de chaque fil sur un bus, la part de la somme de consommation pour plusieurs bits de poids faible par rapport à la consommation totale ainsi que le pourcentage d'apparition de chaque classe de transition du tableau 1.4 pour un flot de données de type image et un bus de 8 bits.

Sur cette figure, la consommation (pour un flot de type données) se situe essentiellement sur

Facteur de délai g du fil victime	Pourcentage d'appartition de la classe de transition
$g = 0$	66.9%
$g = 1$	2.4%
$g = 1 + r$	6.2%
$g = 1 + 2r$	12.7%
$g = 1 + 3r$	10.8%
$g = 1 + 4r$	1.0%

TAB. 3.7 – Pourcentage d'appartition de chacune des classes de transitions du tableau 1.4 pour un flot de données de type image et un bus de 8 bits.

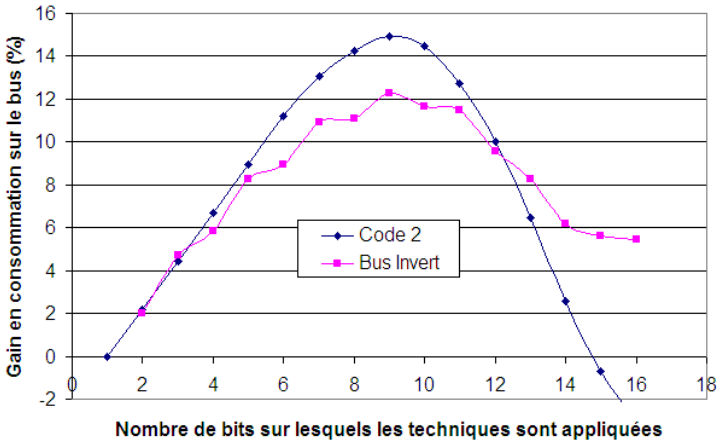


FIG. 3.18 – Pourcentage de réduction de la consommation sur le bus pour les techniques du *Bus Invert* et du *Code 2* en fonction du nombre de bits sur lesquels elles sont appliquées.

les bits de poids faible (déjà plus de 50% de la consommation pour les trois derniers bits). Par conséquent, appliquer les techniques d'optimisation sur tout le bus (et notamment sur les *MSB*) aura un effet très amoindri puisque l'activité des *MSB* est faible (voir le profil d'activité établi dans [LR95]). Pour illustrer ce phénomène, la figure 3.18 présente pour le *Bus Invert* et le *Code 2* le pourcentage de réduction de consommation sur le bus (ratio entre la consommation sur le bus sans codage et avec codage) en fonction du nombre de bits sur lequel les techniques sont appliquées. Au delà d'un certain nombre de bits, le gain en consommation redescend puisque utiliser les techniques sur des bits corrélés (i.e. les *MSB*) aura un effet quasi nul et de ce fait le surcoût en consommation des codecs sera prépondérant. Il est important de noter que d'autres flots de type donnée (musique, parole) ont le même comportement.
De plus, un autre résultat intéressant du tableau 3.7 est que les techniques ayant pour but de supprimer les pires cas du tableau 1.4 (i.e. $1+4r$ et/ou $1+3r$) ne suppriment qu'une partie infime de la totalité des transitions (seulement 1% pour celles qui éliminent les transitions de type $1+4r$ par exemple).
La suppression de ces transitions va certes diminuer le temps de propagation sur le bus, mais ne va pas forcément diminuer la consommation. En effet, ce type de transition sera remplacé par d'autres transitions dont la consommation peut éventuellement être plus importante (si on transforme une transition montante en une transition descendante par exemple).

3.3 Bilan

Dans ce chapitre, un état de l'art des différentes techniques d'optimisation des performances des interconnexions à différents niveaux d'abstraction a été présenté. Nous avons vu que les techniques d'optimisation peuvent cibler préférentiellement soit le bus d'adresses ou le bus de données, soit être applicables aux deux types de bus.
Une synthèse des points forts et des points faibles de ces techniques a également été faite afin de montrer qu'elles peuvent optimiser soit le délai soit la consommation sur le bus, voire les deux pour les plus efficaces d'entre elles.
Nous avons également montré que la plupart de ces techniques se basant sur le classement des transitions du tableau 1.4 permettent effectivement de réduire la consommation, mais ne s'attaquent pas forcément aux transitions les plus pénalisantes en termes de consommation (les transitions descendantes où la capacité présentée par le fil victime est la plus importante).
De plus, beaucoup de ces techniques ont un surcoût matériel non négligeable (ajoutant un surcoût de consommation important), dû au codage des données.
Pour que les méthodes proposées soient vraiment efficaces, il faut que le surcoût de consommation dû aux codecs ne soit pas supérieur à la réduction de consommation apportée sur le bus. Ceci n'est malheureusement pas le cas pour certaines des méthodes présentées dans ce chapitre, car la complexité des codecs est souvent importante (banc de registres, additionneurs, multiplexeurs...). Dans [KNM04], il a été démontré que certaines des techniques ne sont plus efficaces pour les longueurs d'interconnexions actuelles dans les SoC, ceci étant dû au surcoût en consommation des codecs.
Nous avons également montré ici que beaucoup de techniques s'appliquent sur le bus complet alors que la plus grande partie de la consommation se situe là où l'activité est la plus importante (à savoir sur les bits de poids faible).
Le prochain chapitre aura pour objectif de présenter de nouvelles solutions basées sur les conclusions qui viennent d'être établies, à savoir :

- ne pas se focaliser uniquement sur les transitions de type $1+3r$ et $1+4r$ car ce ne sont pas forcément celles qui apparaissent le plus souvent ;
- travailler là où l'activité est la plus importante sur le bus (i.e. les LSB) car ce sont les lignes les plus consommatrices ;
- éviter le plus possible les transitions descendantes. Un point clé pour l'optimisation peut être de coder les données de telle manière que les transitions descendantes sur le bus apparaissent avec la plus faible capacité possible pour le fil victime et donc consomment le moins d'énergie ;
- avoir un surcoût matériel des codecs le plus faible possible et donc avoir des techniques les plus simples possibles.

Chapitre 4

Proposition de techniques d'optimisation au niveau architectural

Sommaire

Résumé

Ce quatrième chapitre présente les techniques d'optimisation au niveau architectural auxquelles nous avons abouti en nous basant sur les pistes d'optimisation du chapitre précédent. Ces techniques (dont une est brevetée : *Spatial Switching*) ont pour particularité de nécessiter un surcoût matériel relativement faible. En effet, nombre de méthodes présentées dans la littérature ont un surcoût matériel assez important, en particulier dû aux codeurs et décodeurs. Ces codecs engendrent un surcoût en consommation bien souvent supérieur à la réduction apportée sur le bus pour des longueurs d'interconnexions usuelles dans les *SoC* actuels. Nos résultats expérimentaux sur les techniques d'optimisation montrent des gains en consommation pouvant atteindre 13% de consommation d'énergie pour un bus de 5mm en 65nm. Ces résultats incluent bien évidemment la consommation due aux codecs. Ces gains augmentent encore avec les futurs sauts technologiques ainsi qu'avec l'augmentation de la longueur du bus.

4.1 Introduction : rappel des principes d'optimisation

Comme nous l'avons vu dans le chapitre précédent, il devient difficile avec les contraintes actuelles (longueurs d'interconnexions, complexité d'architecture de codecs ...) de proposer des solutions de réduction de la consommation des interconnexions à la fois simples et performantes. C'est pourquoi une analyse des critères d'optimisation de la consommation a été proposée. Il en est ressorti que très peu de techniques présentées dans la littérature sont encore applicables ; cela étant dû au fait qu'elles ne s'attaquent pas forcément aux transitions les plus pénalisantes en terme de consommation ainsi qu'à la complexité de leurs codecs. Nous allons dans ce chapitre présenter plusieurs techniques d'optimisation au niveau architectural en adéquation avec les pistes d'optimisation proposées dans le chapitre précédent qui sont :

- ne pas se focaliser uniquement sur les transitions de type $1+3r$ et $1+4r$ parce que elles ne sont pas forcément les plus significatives dans le nombre total des transitions ;
- travailler là où l'activité est la plus importante sur le bus (ie : les LSB) car ce sont les lignes les plus consommatrices ;
- éviter le plus possible les transitions descendantes. Un point clé pour l'optimisation est de coder les données de telle manière que les transitions descendantes sur le bus apparaissent avec la plus faible capacité possible pour le fil victime et donc consomme le moins d'énergie ;
- avoir un surcoût matériel des codecs le plus faible possible impliquant donc des techniques simples.

4.2 Un cycle, une transition au maximum

4.2.1 Principe de la méthode

La technique présentée est basée sur une architecture de codage classique, telle que représentée sur la figure 3.8, consistant en un codeur à l'entrée du bus et en un décodeur à la sortie du bus. Le codeur a pour but de transformer les n bits de la donnée en entrée en m bits de données sur le bus codé (avec $n \leqslant m$) ; le décodeur réalise l'opération inverse afin de récupérer les données originales.

Quand la technique est utilisée, le bus est divisé en groupes de deux fils. Chaque paire de fils, est codée sur trois fils en sortie du bloc de codage tel qu'illustré sur la figure 4.1. Par conséquent pour un bus de taille n, un total de $\frac{n}{2}$ fils supplémentaires est nécessaire.

L'objectif du codage est de diminuer le plus possible l'activité des bits de poids faible avec comme seconde contrainte de faire en sorte que le bus présente la plus faible capacité de *crosstalk* possible. Le point clé de la technique, dans l'objectif de diminuer l'activité, est de faire en sorte que sur les trois fils en sortie du bloc de codage, il n'y ait au plus qu'un seul fil au maximum qui effectue une transition par cycle d'horloge. De cette manière, la diminution de l'activité entraîne une diminution de la consommation. Ceci permet également une accélération de la propagation des données sur le bus car il ne reste au pire cas que des transitions de type $1+2r$ sur les fils codés par la technique.

Définissons :

- $A_j(t_i)$ comme étant la donnée non codée sur le fil j au cycle d'horloge t_i avec $j \in [0,1]$;

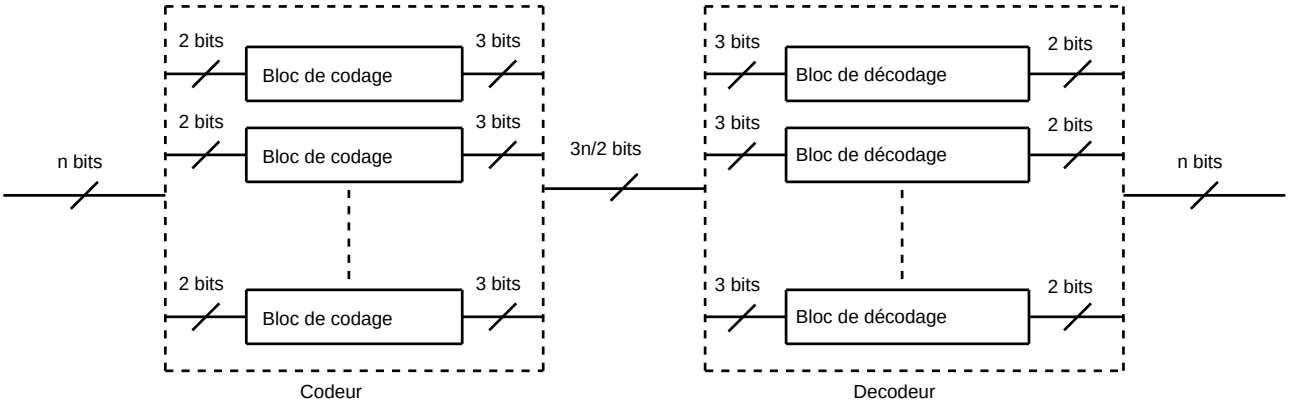


FIG. 4.1 – Architecture macro-blocs des codeurs et décodeurs utilisés par la technique.

- $B_k(t_i)$ comme étant la donnée codée sur le fil k au cycle d'horloge t_i avec $k \in [0, 1, 2]$;
- $B_k(t_{i-1})$ comme étant la donnée sur le fil k au cycle d'horloge t_{i-1} avec $k \in [0, 1, 2]$;

Le principe du codage est de faire en sorte que les deux valeurs consécutives codées $B(t_i)$ et $B(t_{i-1})$ ($B(t_i)$ étant le triplet composé de $B_0(t_i), B_1(t_i), B_2(t_i)$ et $B(t_{i-1})$ celui composé de $B_0(t_{i-1}), B_1(t_{i-1}), B_2(t_{i-1})$) aient le moins de différence possible (distance de *Hamming* minimum ; ici nous nous autorisons 1 au maximum). C'est à dire que $B(t_i) \oplus B(t_{i-1})$ ne doit contenir au maximum qu'un seul 1 (i.e. une seule transition entre les deux données codées). Dans ce cas, $B(t_i) \oplus B(t_{i-1})$ peut prendre pour valeurs 000, 100, 010 et 001 (101 étant exclu car cela signifie qu'il y a deux transitions). Ces quatre valeurs sont donc suffisantes pour coder les quatre valeurs que peuvent prendre les deux fils en entrée ($A_0(t_i)$ et $A_1(t_i)$). La génération des valeurs binaires $B_k(t_i)$ composant le triplet $B(t_i)$ (pour $k \in [0, 1, 2]$) se fait donc de la manière suivante :

$$\begin{aligned} B_0(t_i) &= B_0(t_{i-1}) \oplus [A_1(t_i)A_0(t_i)] \\ B_1(t_i) &= B_1(t_{i-1}) \oplus \left[\overline{A_1(t_i)}A_0(t_i)\right] \\ B_2(t_i) &= B_2(t_{i-1}) \oplus \left[A_1(t_i)\overline{A_0(t_i)}\right] \end{aligned} \tag{4.1}$$

Le bloc de codage de la figure 4.2 illustre l'architecture associée à l'équation précédente. Du côté du décodeur, le même principe est appliqué afin de retrouver les données originales :

$$\begin{aligned} C_0(t_i) &= B_0(t_{i-1}) \oplus B_0(t_i) \\ C_1(t_i) &= B_1(t_{i-1}) \oplus B_1(t_i) \\ C_2(t_i) &= B_2(t_{i-1}) \oplus B_2(t_i) \end{aligned} \tag{4.2}$$

$$\begin{aligned} C_0(t_i) &= B_0(t_{i-1}) \oplus B_0(t_{i-1}) \oplus [A_1(t_i)A_0(t_i)] \\ C_1(t_i) &= B_0(t_{i-1}) \oplus B_1(t_{i-1}) \oplus \left[\overline{A_1(t_i)}A_0(t_i)\right] \\ C_2(t_i) &= B_0(t_{i-1}) \oplus B_2(t_{i-1}) \oplus \left[A_1(t_i)\overline{A_0(t_i)}\right] \end{aligned} \tag{4.3}$$

$$\begin{aligned} C_0(t_i) &= A_1(t_i)A_0(t_i) \\ C_1(t_i) &= \overline{A_1(t_i)}A_0(t_i) \\ C_2(t_i) &= A_1(t_i)\overline{A_0(t_i)} \end{aligned} \tag{4.4}$$

Soit les deux bits décodés en sortie du bus :

$$D_0(t_i) = C_0(t_i) + C_1(t_i) = A_0(t_i)$$
$$D_1(t_i) = C_0(t_i) + C_2(t_i) = A_1(t_i) \tag{4.5}$$
$$\tag{4.6}$$

Le bloc de décodage de la figure 4.2 illustre l'architecture associée aux équations précédentes. La figure 4.3 illustre le fonctionnement de la technique sur un flot de données simples. Le flot de données original a une activité moyenne de 62.5%. En utilisant la technique, le flot de données codées voit son activité réduite à 33%. La section suivante va présenter les résultats expérimentaux sur l'optimisation de la consommation obtenue par notre première technique d'optimisation, en tenant compte de la variation de différents paramètres technologiques.

4.2.2 Résultats expérimentaux

Pour les simulations, le modèle du bus utilisé est un modèle distribué Π_3 incluant les capacités de *crosstalk* comme nous l'avons présenté dans le premier chapitre. Les résultats expérimentaux ont été obtenus en utilisant un simulateur SPICE (ELDO v5.7), pour différentes technologies (130nm, 90nm et 65nm), en considérant les stimulis de type données (image, musique, parole).
Chaque technologie possède un certain nombre de couches de métal : 5 pour la 130nm, 6 pour la 90nm et 7 pour la 65nm. Les simulations couvrent toutes les couches de métal, des plus basses (utilisées pour de courtes interconnexions) jusqu'au plus hautes (utilisées pour de long bus, ce qui nous intéresse ici plus particulièrement).
Comme le temps de propagation sur les interconnexions peut être une donnée critique, les expérimentations considèrent également les lignes bufferisées et non bufferisées.
Comme le temps de propagation et la consommation sont parmi les points clés dans la conception des *SoC* actuels, et dépendent entre autre de la taille des transistors (largeur W_{MOS}), la largeur des transistors utilisés dans les codecs est un paramètre supplémentaire pour les simulations. W_{MOS} peut varier ici de deux fois la longueur de la technologie (2λ, cette taille est la taille minimum de *design* fixée par les valeurs des *design kits*) à n fois 2λ (pour les expérimentations la borne haute s'arrête à 6). Les résultats de consommation présentés dans le tableau 4.1 tiennent évidemment compte du surcoût de consommation apporté par les codecs.
Premièrement, l'analyse des résultats permet de remarquer que l'efficacité de la technique augmente lors des sauts technologiques, ce qui est un point très important pour la réduction de la consommation dans les technologies actuelles et futures.
Deuxièmement, nous pouvons remarquer que la technique est tout de suite efficace pour les couches de métal les plus basses et devient de plus en plus efficace en montant dans les couches de métal ; couches hautes, réservées plus particulièrement pour les bus, sur lesquelles le gain sera encore plus

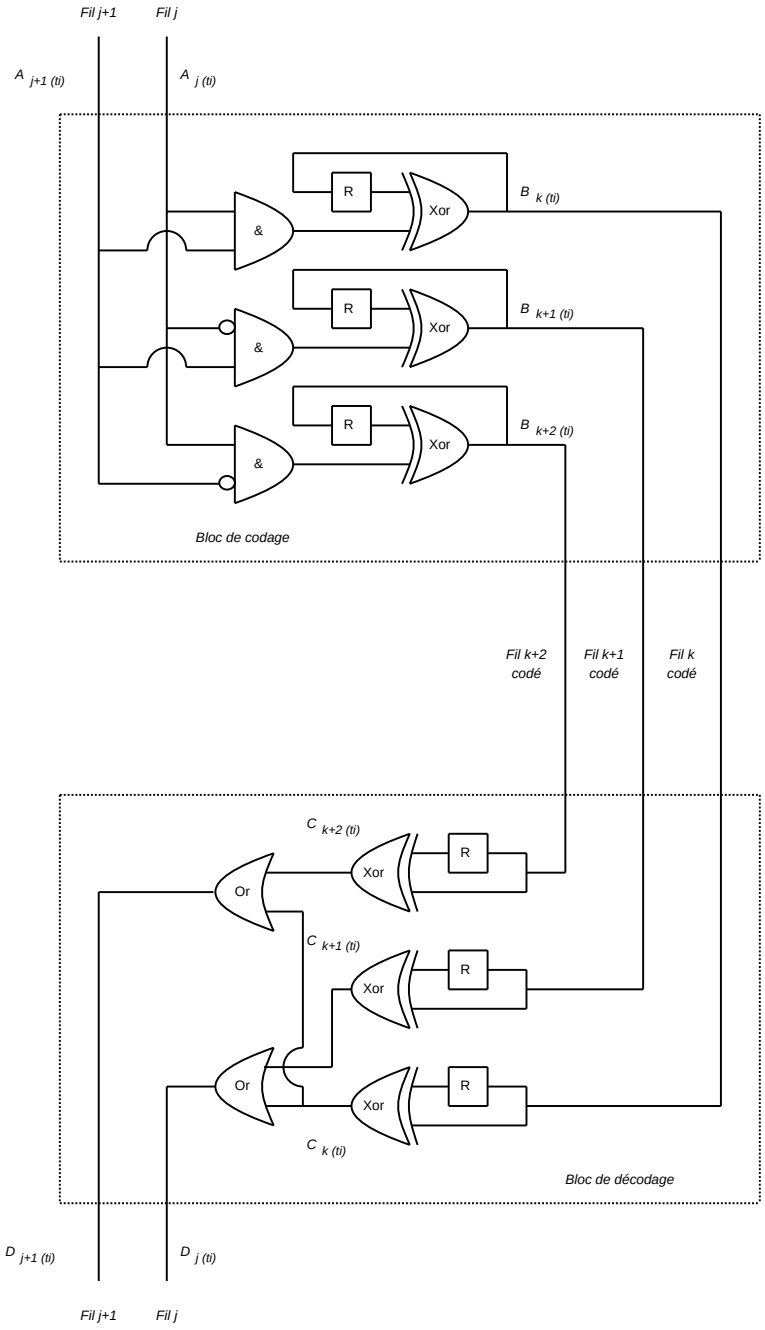


FIG. 4.2 – Architecture interne des macro-blocs *bloc de codage* et *bloc de décodage* des codecs présentés à la figure 4.1.

	130nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	1.14	-1.04	-14.71	-38.50
6λ	-1.15	-5.63	-21.60	-47.68
8λ	-3.53	-10.39	-28.73	-57.19
10λ	-5.91	-15.15	-35.88	-66.72
12λ	-8.35	-20.03	-43.20	-76.48

	90nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.14	2.83	-9.22	-30.87
6λ	1.48	-0.49	-14.19	-37.50
8λ	0.32	-2.82	-17.69	-42.16
10λ	-0.87	-5.19	-21.25	-46.90
12λ	-2.06	-7.58	-24.82	-51.67

	65nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.23	5.28	-6.46	-27.58
6λ	3.57	3.95	-8.45	-30.23
8λ	2.91	2.63	-10.42	-32.87
10λ	2.23	1.28	-12.46	-35.58
12λ	1.54	-0.11	-14.54	-38.36

	130nm / Layer 5 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.22	0.94	-11.17	-32.65
6λ	0.42	-2.67	-16.60	-39.88
8λ	-1.41	-6.32	-22.07	-47.18
10λ	-3.28	-10.06	-27.68	-54.66
12λ	-5.17	-13.84	-33.35	-62.21

	90nm / Layer 6 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.12	2.76	-9.65	-31.62
6λ	1.55	-0.38	-14.37	-37.90
8λ	0.45	-2.58	-17.67	-42.30
10λ	-0.69	-4.85	-21.08	-46.85
12λ	-1.80	-7.08	-24.42	-51.31

	65nm / Layer 7 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.39	5.54	-6.01	-27.02
6λ	3.74	4.24	-7.97	-29.64
8λ	3.10	2.95	-9.90	-32.21
10λ	2.44	1.63	-11.88	-34.85
12λ	1.76	0.28	-13.91	-37.55

	130nm / Layer 5 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.35	3.13	-8.06	-29.21
6λ	2.19	0.82	-11.53	-33.84
8λ	1.03	-1.51	-15.03	-38.50
10λ	-0.16	-3.89	-18.60	-43.27
12λ	-1.35	-6.26	-22.15	-48.00

	90nm / Layer 6 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.65	4.20	-7.65	-28.81
6λ	2.64	2.17	-10.70	-32.87
8λ	1.96	0.81	-12.73	-35.58
10λ	1.29	-0.53	-14.74	-38.26
12λ	0.59	-1.93	-16.84	-41.06

	65nm / Layer 7 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.83	5.13	-5.33	-25.03
6λ	3.43	4.33	-6.53	-26.63
8λ	3.03	3.55	-7.71	-28.20
10λ	2.63	2.75	-8.91	-29.80
12λ	2.22	1.93	-10.14	-31.44

TAB. 4.1 – Variation de la consommation sur le bus (incluant le surcoût de consommation dû aux codecs) exprimée en % pour différentes technologies, couches de métal, longueurs de bus, tailles de transistors et nombre de bits sur lesquels la technique est appliquée (de 2 *LSB* à 8 *LSB*). Quand un gain en consommation est observé, les valeurs sont positives, dans le cas contraire les valeurs sont négatives (cases grisées) (i.e. ceci étant dû au fait que la consommation des codecs dépasse la réduction en consommation sur le bus).

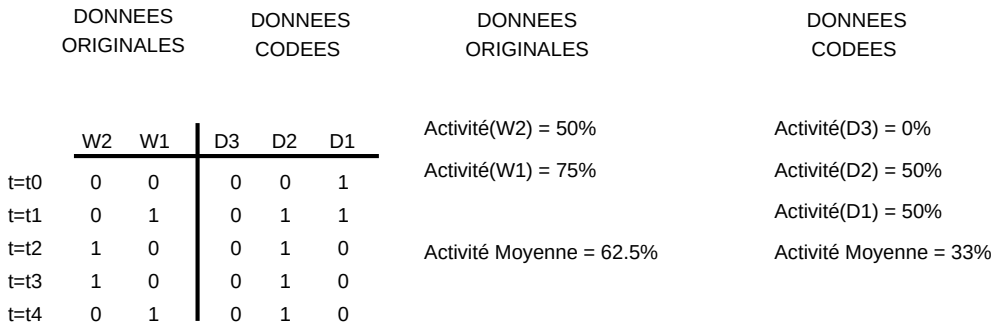


FIG. 4.3 – Exemple de la technique appliquée sur un flot de données, où W_i et D_i représentent le i^{eme} fil respectivement non codé et codé.

important.
Troisièmement, les résultats montrent que plus le bus est long, meilleure est la réduction de consommation. Le gain en consommation (pour les longueurs simulées) peut atteindre 5.5%, ce gain augmentant encore si la longueur augmente. Dans les *SoC* actuels, les bus peuvent varier en moyenne de 1 à 7mm pour des bus très longs. Par conséquent, pour cette plage de variation de longueur, la technique peut apporter une réduction de consommation assez importante.
Quatrièmement, comme souligné dans le chapitre précédent, il existe une valeur optimale pour le nombre de bits sur lesquels la technique peut être appliquée. Les résultats du tableau 4.1 montrent que la technique permet d'obtenir un accroissement des gains en consommation lorsqu'elle est appliquée sur les 2^{eme} et 4^{eme} bits les moins significatifs. Les gains commencent à décroître pour les 6^{eme} et 8^{eme} bits les moins significatifs. Ceci est dû au fait que les *MSB* sont plus corrélés temporellement entre eux que les *LSB*, il y a donc beaucoup moins de réduction d'activité possible sur ces bits. Par conséquent, les codecs placés sur les *MSB* engendrent un surcoût énergétique (différence entre le coût des codecs et la réduction apportée sur le bus).
Nous venons de présenter une technique basée ici sur une réduction de l'activité des lignes les plus consommatrices. Dans une seconde approche, nous allons maintenant nous intéresser aux types de transitions sur ces même lignes. L'objectif de la technique qui va être présentée maintenant, consiste non plus à s'occuper de réduire l'activité mais à coder les transitions de telle sorte qu'elles présentent la plus faible capacité de *crosstalk* sur le bus lors des charges de capacités.

4.3 Le *Spatial Switching*

4.3.1 Principe du *Spatial Switching*

La technique du *Spatial Switching* (technique brevetée [CLSJ08b] que nous avons présentée dans [CLSJ08a]) est basée sur une architecture de codage classique représentée en figure 3.8, consistant en un codeur à l'entrée du bus et en un décodeur à la sortie du bus. Le codeur a pour but de transformer les n bits de la donnée en entrée en m bits de données sur le bus codé (avec $n \leqslant m$) ; le décodeur réalise l'opération inverse afin de récupérer les données originales.
Quand le *Spatial Switching* est utilisé, le bus est divisé en groupes de deux fils. Chaque paire de

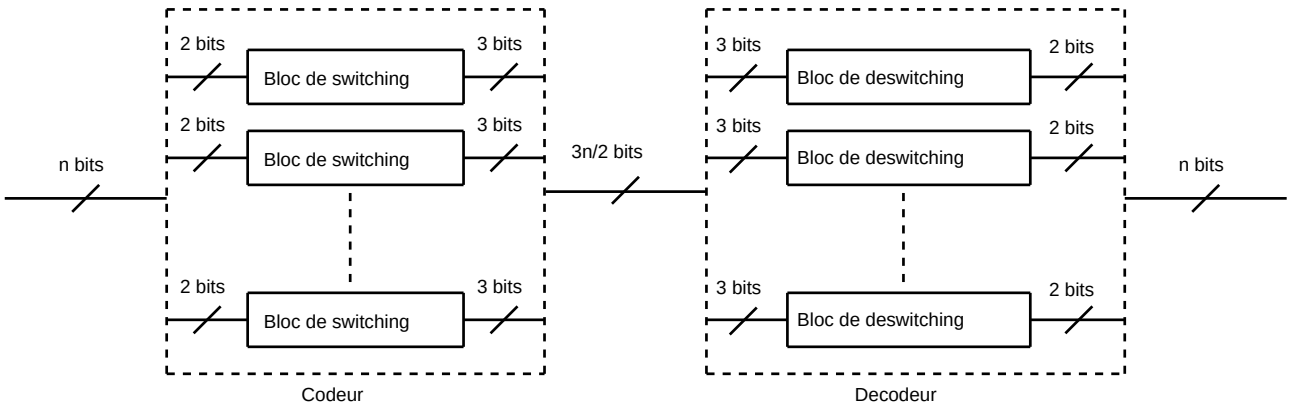


FIG. 4.4 – Architecture macro-blocs des codeurs et décodeurs utilisés pour le *Spatial Switching*.

fil est codée en utilisant le principe décrit ci-après. Pour chaque paire de fils, un fil de contrôle supplémentaire (noté *Switch*) est ajouté en sortie du bloc de codage tel qu'illustré sur la figure 4.4. Par conséquent, pour un bus de taille n, un total de $\frac{n}{2}$ fils supplémentaires est nécessaire. Le principe du codage est de remplacer les transitions les plus consommatrices sur le bus par d'autres moins consommatrices de telle sorte que le bus présente la plus faible capacité de *cross-talk* possible. Le point clé de la technique est de détecter toutes les transitions croisées sur des fils adjacents. Ainsi, éliminer les transitions croisées (une transition croisée contient forcément une transition descendante) permet de supprimer les transitions les plus consommatrices telles que référencées dans le tableau 3.5. Eliminer ces transitions permet également une accélération de la propagation des données sur le bus car il ne reste plus qu'au pire cas des transitions de type $1+2r$ sur les fils codés.

De plus, comme nous l'avons vu dans la section 1.3.2 sur le *crosstalk*, le fait d'éliminer les transitions croisées permet de diminuer le bruit généré sur les interconnexions. Par conséquent, la technique du *Spatial Switching* permet également de rendre la transmission plus fiable en diminuant potentiellement le taux d'erreur binaire sur les lignes par la suppression de ce type de transitions.

Dans le cas où une transition croisée est détectée sur deux fils adjacents, les chemins empruntés par les données sont alors inversés (i.e. soit deux fils 1 et 2 ; si une transition croisée est détectée, la donnée qui devait initialement transiter sur le fil 1, transitera sur le fil 2. Il en va de même pour la donnée devant initialement transiter sur le fil 2). Un signal est également positionné sur le fil de contrôle afin de permettre au décodeur de rediriger correctement les données en sortie du bus. L'algorithme de principe utilisé dans le *Spatial Switching* va maintenant être introduit. Définissons :

- $D_j(t_i)$ comme étant la donnée sur le fil j au cycle d'horloge t_i ;
- $D_j(t_{i-1})$ comme étant la donnée sur le fil j au cycle d'horloge t_{i-1} ;
- $D_{j+1}(t_i)$ comme étant la donnée sur le fil *j+1* au cycle d'horloge t_i, et ;
- $D_{j+1}(t_{i-1})$ comme étant la donnée sur le fil *j+1* au cycle d'horloge t_{i-1}.

L'algorithme utilisé dans le *Spatial Switching* peut alors être exprimé comme dans le tableau 4.2 et dans le tableau 4.3 sous forme d'équations logiques. Une transition croisée est détectée quand

SI	ALORS	SINON
$[D_j(t_i) \neq D_{j+1}(t_i)]$ ET	$D_j(t_i) = D_{j+1}(t_i)$	$D_j(t_i) = D_j(t_i)$
$[D_j(t_i) \neq D_j(t_{i-1})]$ ET	$D_{j+1}(t_i) = D_j(t_i)$	$D_{j+1}(t_i) = D_{j+1}(t_i)$
$[D_{j+1}(t_i) \neq D_{j+1}(t_{i-1})]$	$\overline{Switch} = 0$	$\overline{Switch} = 1$

TAB. 4.2 – Algorithme utilisé dans le *Spatial Switching*.

SI	ALORS	SINON
$[D_j(t_i) \oplus D_{j+1}(t_i)]$ AND	$D_j(t_i) = D_{j+1}(t_i)$	$D_j(t_i) = D_j(t_i)$
$[D_j(t_i) \oplus D_j(t_{i-1})]$ AND	$D_{j+1}(t_i) = D_j(t_i)$	$D_{j+1}(t_i) = D_{j+1}(t_i)$
$[D_{j+1}(t_i) \oplus D_{j+1}(t_{i-1})]$ $= 0$	$\overline{Switch} = 0$	$\overline{Switch} = 1$

TAB. 4.3 – Equations logiques utilisées dans le *Spatial Switching* (i.e. $\oplus$ représente un ou exclusif et AND un et).

les données sur les deux fils voisins sont différentes au cycle d'horloge t_i et au cycle d'horloge t_{i-1}, et également quand les données sur le même fil sont différentes au cycle d'horloge t_i et au cycle d'horloge t_{i-1}. Afin de réaliser cette détection, des portes *XOR* sont utilisées. Si toutes les sorties de ces portes sont au niveau logique 1 (i.e. c'est-à-dire qu'il y a eu une transition croisée), le signal de contrôle *Switch* est alors positionné au niveau logique 0. La génération du signal de contrôle *Switch* est illustré à la figure 4.5 dans le bloc nommé *Bloc de détection des transitions croisées*. La figure 4.5 présente également le bloc de routage des données sur les deux fils voisins. Ce bloc consiste en deux multiplexeurs 2 vers 1 dont les sorties dépendent évidemment du niveau logique du signal de contrôle *Switch*. En fin de bus, un autre bloc de routage est utilisé (également piloté par le signal de contrôle *Switch*) pour rediriger les données codées afin de récupérer les données originales sur les deux fils voisins.
La figure 4.6 illustre le fonctionnement du *Spatial Switching* sur un flot de données simples. Le flot de données original contient deux transitions croisées. En utilisant le *Spatial Switching*, le flot de données codées ne contient plus aucune transition croisée. La section suivante va présenter les résultats expérimentaux sur l'optimisation de la consommation obtenus en tenant compte de la variation de différents paramètres technologiques.

4.3.2 Résultats expérimentaux

Pour les simulations, les conditions d'expérimentation sont rigoureusement les mêmes que celles utilisées dans la présentation des résultats expérimentaux de la technique précédente. Les résultats de consommation présentés dans le tableau 4.4 tiennent évidemment compte du surcoût de consommation apporté par les codecs.
Premièrement, l'analyse des résultats permet de remarquer que l'efficacité du *Spatial Switching* augmente lors des sauts technologiques, ce qui est un point très important pour la réduction de

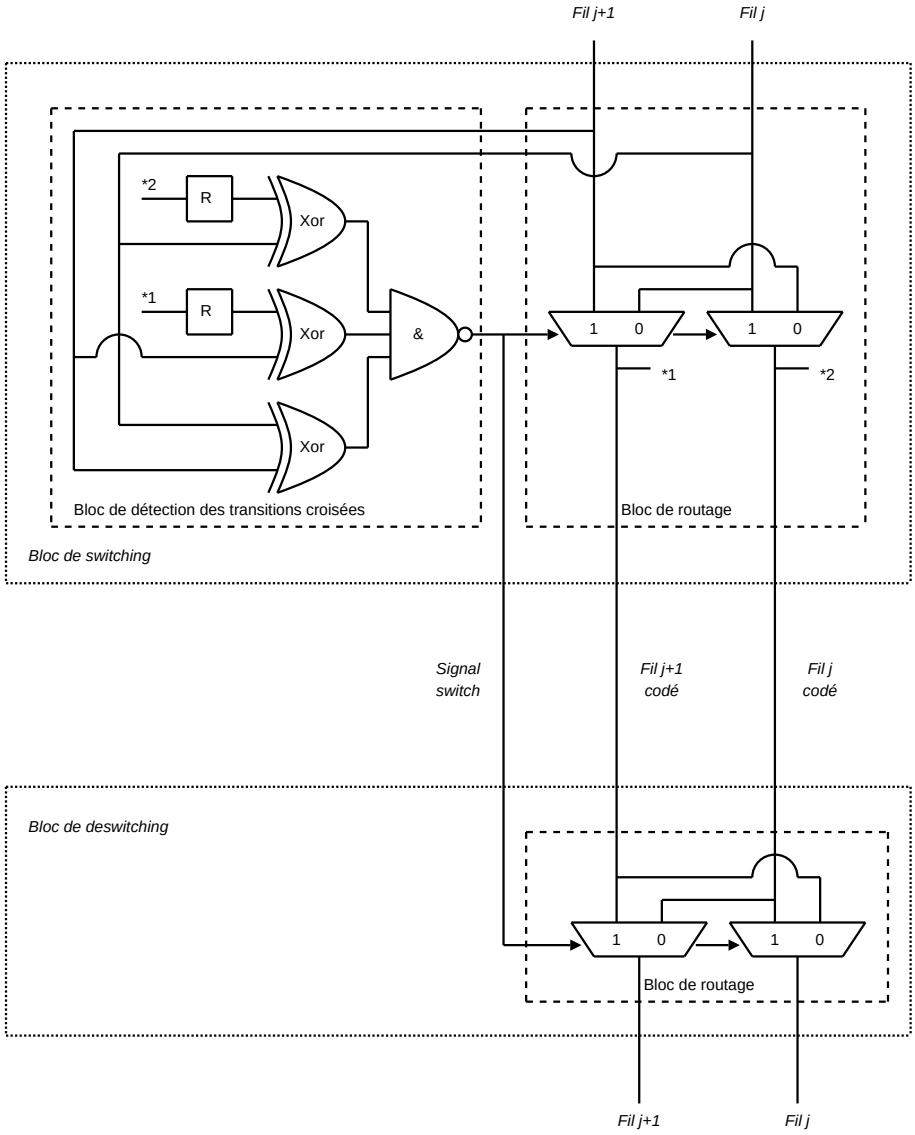


FIG. 4.5 – Architecture interne des macro-blocs *bloc de switching* et *bloc de deswitching* des codecs présentés à la figure 4.4.

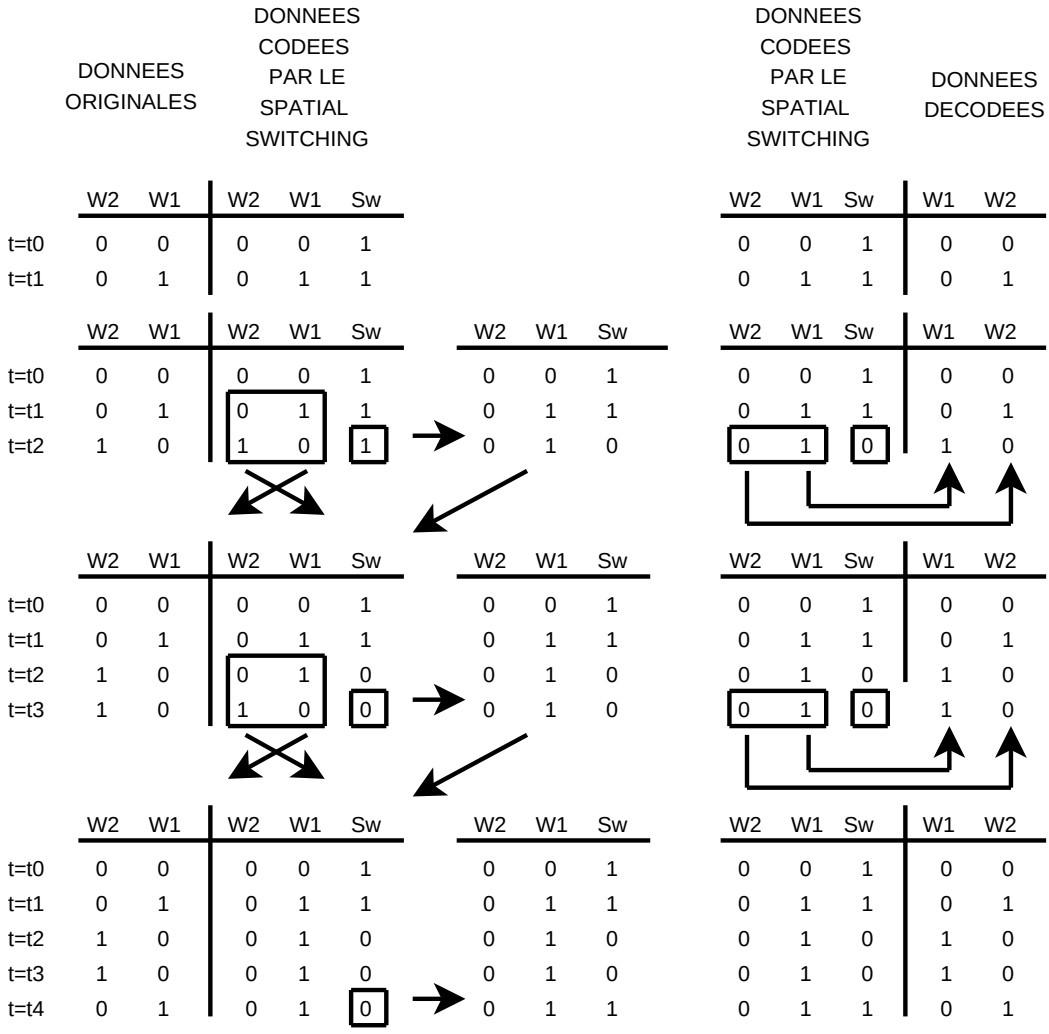


FIG. 4.6 – Exemple du *Spatial Switching* appliqué sur un flot de données, où W_i represente le i^{eme} fil et Sw le signal de contrôle *Switch*.

	130nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.78	4.06	5.20	2.64
6λ	1.09	0.68	0.13	-4.12
8λ	-0.52	-2.55	-4.70	-10.56
10λ	-2.11	-5.74	-9.49	-16.95
12λ	-3.75	-9.02	-14.41	-23.51

	90nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.42	7.52	10.44	10.00
6λ	3.11	4.88	6.49	4.73
8λ	2.09	2.86	3.45	0.68
10λ	1.85	2.38	2.73	-0.28
12λ	0.41	-0.52	-1.61	-6.07

	65nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.06	7.11	10.19	10.20
6λ	3.63	6.25	8.90	8.48
8λ	3.18	5.35	7.54	6.67
10λ	2.71	4.41	6.13	4.79
12λ	2.20	3.38	4.60	2.74

	130nm / Layer 5 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.66	5.77	7.54	5.80
6λ	2.26	2.96	3.32	0.18
8λ	0.94	0.33	-0.62	-5.08
10λ	-0.31	-2.18	-4.39	-10.09
12λ	-1.58	-4.71	-8.19	-15.16

	90nm / Layer 6 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.39	7.74	10.36	9.95
6λ	3.18	5.05	6.73	5.10
8λ	2.21	3.11	3.82	1.22
10λ	1.30	1.30	1.10	-2.40
12λ	0.54	-0.24	-1.20	-5.47

	65nm / Layer 7 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.15	7.26	10.46	10.50
6λ	3.74	6.44	9.22	8.86
8λ	3.31	5.57	7.92	7.12
10λ	2.84	4.64	6.53	5.26
12λ	2.37	3.70	5.12	3.38

	130nm / Layer 5 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.84	8.19	11.19	10.60
6λ	4.04	6.59	8.79	7.40
8λ	3.14	4.79	6.10	3.81
10λ	2.23	2.97	3.37	0.17
12λ	1.36	1.22	0.74	-3.33

	90nm / Layer 6 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.95	8.35	11.75	12.05
6λ	4.31	7.06	9.82	9.47
8λ	3.78	6.01	8.25	7.38
10λ	3.22	4.89	6.56	5.13
12λ	2.60	3.66	4.71	2.66

	65nm / Layer 7 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	4.11	7.38	10.64	10.55
6λ	3.87	6.90	9.92	9.59
8λ	3.62	6.40	9.18	8.60
10λ	3.36	5.88	8.40	7.57
12λ	3.10	5.38	7.61	6.51

TAB. 4.4 – Variation de la consommation sur le bus (incluant le surcoût de consommation dû aux codecs) exprimée en % pour différentes technologies, couches de métal, longueurs de bus, tailles de transistors et nombre de bits sur lesquels le *Spatial Switching* est appliqué (de 2 *LSB* à 8 *LSB*) pour un flot de données de type image. Quand un gain en consommation est observé, les valeurs sont positives, dans le cas contraire les valeurs sont négatives (cases grisées) (i.e. ceci étant dû au fait que la consommation des codecs dépasse la réduction en consommation sur le bus).

la consommation dans les technologies actuelles et futures.
Deuxièmement, nous pouvons remarquer que le *Spatial Switching* est même efficace pour les couches de métal les plus basses et devient de plus en plus efficace avec l'élévation dans les couches de métal ; c'est sur les couches hautes, réservées plus particulièrement pour les bus que le gain sera encore plus important.
Troisièmement, les résultats montrent que plus le bus est long, meilleure est la réduction de consommation. Le gain en consommation (pour les longueurs simulées) peut atteindre 12% pour un bus de 5mm, ce gain augmente encore si la longueur augmente. Dans les *SoC* actuels, les bus peuvent varier en moyenne de 1 à 7mm pour des bus très longs. Par conséquent, pour cette plage de variation de longueur, le *Spatial Switching* peut apporter une réduction de consommation conséquente.
Les longueurs des bus, utilisés pour les résultats expérimentaux des techniques d'optimisation dans la littérature, ne sont pas toujours très réalistes. Par exemple, la technique utilisée dans [KIA99] montre un gain en consommation pour des longueurs de bus atteignant 7.5cm. De plus, les résultats présentés dans [KNM04] montrent que beaucoup de techniques commencent à être efficaces pour de très long bus (à partir de 2cm pour le *Bus Invert* par exemple), ceci étant principalement dû au surcoût de consommation apporté par les codecs. Il est également important de noter que certaines des techniques présentées ([BMM+97], [PPS06] par exemple) ne prennent pas en compte l'évaluation de consommation des codecs lors de la présentation des résultats expérimentaux.
Quatrièmement, comme souligné dans le chapitre précédent, il existe une valeur optimale du nombre de bits sur lequel la technique peut être appliquée. Le *Spatial Switching* a pour but d'éliminer les transitions croisées, or, ce type de transition a une probabilité d'apparition plus forte sur les fils dont l'activité est importante (i.e. les *LSB*). Les résultats du tableau 4.4 montrent que la technique permet d'obtenir un accroissement des gains en consommation lorsqu'elle est appliquée sur les 2, 4 et 6 *LSB*. Les gains commencent à décroître pour les 8 *LSB*. Ceci est dû au fait que les *MSB* sont plus corrélés entre eux que les *LSB* ; il y a donc moins de transitions croisées sur ces premiers. Par conséquent, les codecs placés sur les 2 *MSB* consomment plus que la réduction de consommation qu'ils apportent. Cette valeur optimale du nombre de bits peut être différente en fonction du type de données qui circulent sur le bus. Ici les résultats présentés dans le tableau 4.4 sont obtenus pour le transfert d'un flot de données de type image. Or dans le cadre d'autres applications du *TDSI* (génération des données en sortie d'un papillon de *FFT*, entrelacement temporel des données ...) il se peut que le profil d'activité des données deviennent totalement aléatoire. Les résultats du tableau 4.5 montrent alors que pour ce type de profil d'activité, il est intéressant d'appliquer le *Spatial Switching* sur la totalité du bus.
Cinquièmement, les résultats du tableau 4.6 montrent que, plus les transistors utilisés dans les codecs sont larges et plus la fréquence de fonctionnement maximale atteignable par les codecs est importante. Si la fréquence de fonctionnement n'est pas un paramètre contraignant pour l'application, alors, le meilleur choix à faire pour obtenir les meilleurs gains en consommation sur le bus est d'utiliser des transistors de taille minimale. Dans le cas contraire, un compromis doit être fait dans le choix de la taille des transistors et la fréquence maximale afin d'obtenir les meilleurs résultats dans la réduction de consommation. Notons également que l'utilisation du *Spatial Switching* permet d'obtenir des fréquences de fonctionnement plus élevées sur le bus

	130nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	1.75	2.99	4.23	5.72
6λ	0.60	0.68	0.77	1.12
8λ	-0.50	-1.51	-2.52	-3.27
10λ	-1.59	-3.68	-5.78	-7.72
12λ	-2.71	-5.92	-9.13	-12.09

	90nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.76	5.06	7.52	10.13
6λ	1.88	3.31	4.88	6.61
8λ	1.21	1.96	2.86	3.92
10λ	1.05	1.64	2.38	3.28
12λ	0.08	-0.29	-0.51	-0.58

	65nm / Layer 3 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.57	4.82	7.06	9.47
6λ	2.28	4.24	6.19	8.31
8λ	1.97	3.62	5.27	7.09
10λ	1.66	2.99	4.32	5.82
12λ	1.31	2.30	3.29	4.44

	130nm / Layer 5 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.74	4.70	6.86	9.21
6λ	1.79	2.81	4.02	5.42
8λ	0.91	1.04	1.37	1.89
10λ	0.07	-0.64	-1.16	-1.48
12λ	-0.78	-2.35	-3.72	-4.89

	90nm / Layer 6 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.76	5.23	7.56	10.18
6λ	1.95	3.60	5.12	6.92
8λ	1.30	2.30	3.16	4.41
10λ	0.69	1.08	1.33	1.88
12λ	0.17	0.05	-0.21	-0.18

	65nm / Layer 7 / 3mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.62	4.78	7.09	9.56
6λ	2.35	4.23	6.26	8.45
8λ	2.06	3.64	5.38	7.28
10λ	1.75	3.02	4.45	6.04
12λ	1.43	2.39	3.51	4.78

	130nm / Layer 5 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.26	5.92	8.69	11.71
6λ	2.73	4.86	7.10	9.59
8λ	2.13	3.66	5.31	7.21
10λ	1.53	2.46	3.50	4.79
12λ	0.95	1.29	1.76	2.47

	90nm / Layer 6 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	3.14	5.75	8.45	11.33
6λ	2.71	4.88	7.15	9.59
8λ	2.36	4.18	6.09	8.18
10λ	1.98	3.43	4.96	6.67
12λ	1.56	2.59	3.71	5.01

	65nm / Layer 7 / 5mm			
	2LSB	4LSB	6LSB	8LSB
4λ	2.70	5.12	7.64	10.34
6λ	2.54	4.80	7.16	9.70
8λ	2.38	4.47	6.66	9.04
10λ	2.21	4.13	7.15	8.35
12λ	2.03	3.78	5.62	7.65

TAB. 4.5 – Variation de la consommation sur le bus (incluant le surcoût de consommation dû aux codecs) exprimée en % pour différentes technologies, couches de métal, longueurs de bus, tailles de transistors et nombre de bits sur lesquels le *Spatial Switching* est appliqué (de 2 *LSB* à 8 *LSB*) pour un flot de données aléatoires. Quand un gain en consommation est observé, les valeurs sont positives, dans le cas contraire les valeurs sont négatives (cases grisées) (i.e. ceci étant dû au fait que la consommation des codecs dépasse la réduction en consommation sur le bus).

	130nm / Layer 5			
	1mm	3mm	5mm	7mm
4λ	3.48	2.65	2.16	1.83
6λ	3.94	3.26	2.75	2.39
8λ	4.25	3.68	3.12	2.83
10λ	4.46	3.97	3.46	3.14
12λ	4.62	4.17	3.75	3.38

90nm / Layer 6			
1mm	3mm	5mm	7mm
3.68	2.21	1.58	1.23
3.81	2.73	2.21	1.75
3.85	3.02	2.46	2.08
4.00	3.22	2.75	2.34
4.12	3.37	2.94	2.56

65nm / Layer 7			
1mm	3mm	5mm	7mm
7.93	4.06	2.75	1.52
9.36	5.43	3.85	2.22
10.31	6.52	4.74	2.84
10.86	7.50	5.52	3.41
11.25	8.14	6.20	3.94

TAB. 4.6 – Fréquence de fonctionnement atteignable par les codecs (exprimée en GHz) pour différentes technologies, couches de métal, longueurs de bus et tailles de transistors.

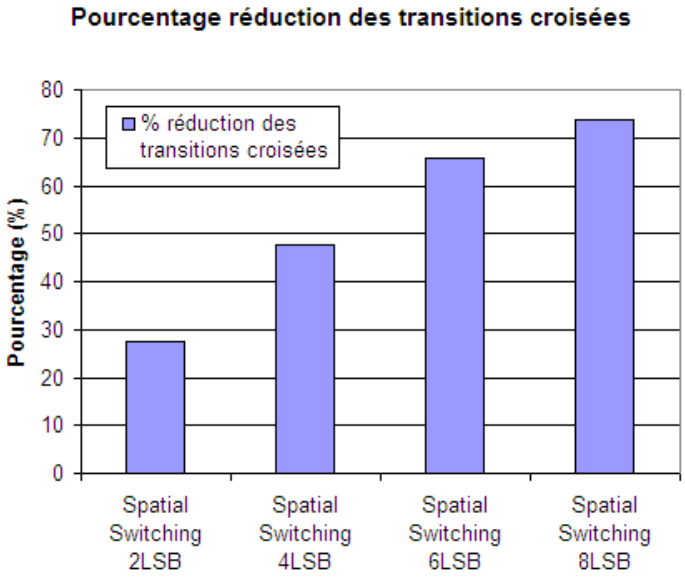


FIG. 4.7 – Pourcentage de transitions croisées éliminées en fonction du nombre de bits sur lequel le *Spatial Switching* est appliqué.

puisque les pires cas de transition de la classification temporelle du tableau 1.4 sont supprimés (i.e. les transitions de type $1 + 3r$ et $1 + 4r$ n'existent plus). Il est également important de noter que la totalité des transitions croisées est certe éliminée sur les fils codés mais il est possible qu'il en reste entre les fils non codés, entre les fils de contrôles supplémentaires et entre les blocs de fils codés. La figure 4.7 illustre le pourcentage de transitions croisées éliminées en fonction du nombre de bits sur lequel le *Spatial Switching* est utilisé. Lorsque le *Spatial Switching* est utilisé sur la totalité du bus, environ 75% des transitions croisées sont éliminées.

Comparaison du *Spatial Switching* et du *Partial Bus Invert* Puisque nous avons vu dans la section 3.1.6 que le *Partial Bus Invert* est la technique proposée dans la littérature qui apporte les meilleurs gains, nous avons donc décidé de comparer les performances de cette technique avec le *Spatial Switching*.

	Cas 1		Cas 2		Transistors		Registres		Fils supplémentaires	
	SS	PBI	SS	PBI	SS	PBI	SS	PBI	SS	PBI
2LSB	6.2	3.9	4.4	<0	48	48	2	2	1	1
4LSB	11.1	10.0	7.5	<0	96	144	4	4	2	1
6LSB	15.8	14.3	10.4	<0	144	240	6	6	3	1
8LSB	17.1	6.8	10.0	<0	192	336	8	8	4	1

TAB. 4.7 – Ce tableau, présente la comparaison des techniques du *Spatial Switching* (SS) et du *Partial Bus Invert* (PBI) en fonction du nombre de bits sur lequel elles sont appliquées. Le Cas 1 représente le gain en consommation sur le bus (en %) sans tenir compte du surcoût de consommation apporté par les codecs. Le Cas 2 quant à lui, présente le gain en consommation sur le bus (en %) en tenant compte du surcoût de consommation apporté par les codecs. La comparaison entre les complexités des deux techniques en termes de nombre de transistors, de registres et de fils supplémentaire est également proposée.

Le tableau 4.7 montre les résultats en termes de gain en consommation pour deux cas de figure en fonction du nombre de bits sur lesquels sont appliquées les deux techniques pour une technologie 90nm sur la couche de métal 3 avec une longueur de bus de 3mm. Le cas 1 montre le gain en consommation sur le bus apporté par les codages, sans le surcoût de consommation dû aux codecs. Le cas 2 montre le gain en consommation sur le bus apporté par les codages, en incluant le surcoût de consommation dû aux codecs.
Dans ce tableau, sont aussi représentées les complexités en termes de nombre de transistors, de registres et de fils supplémentaires en fonction du nombre de bits sur lesquels sont appliquées les deux techniques.
Les résultats de ce tableau montrent, dans un premier temps, que le *Spatial Switching* permet d'obtenir de meilleurs résultats de gain de consommation sur le bus ; ceci sans tenir compte du surcoût de consommation des codecs.
Dans un second temps, en tenant compte du surcoût de consommation des codecs (il a été démontré dans [KNM04] que pour les longueurs de bus que nous simulons ici que le surcoût de consommation des codecs est supérieur à la réduction apportée sur le bus), le *Spatial Switching* permet toujours d'obtenir des gains conséquents alors que ce n'est plus le cas du *Partial Bus Invert*.
Enfin, il est important de remarquer que, à complexité équivalente (en termes du nombre de transistors), le *Spatial Switching* peut s'appliquer sur un plus grand nombre de bits du bus. Ceci est intéressant d'un point de vue de la surface occupée par les codecs, car à nombre de transistors équivalent, notre technique permet d'obtenir encore plus de gain.

4.4 Proposition d'évolution du *Spatial Switching*

Nous allons dans cette section proposer plusieurs pistes d'évolution de la technique du *Spatial Switching*.

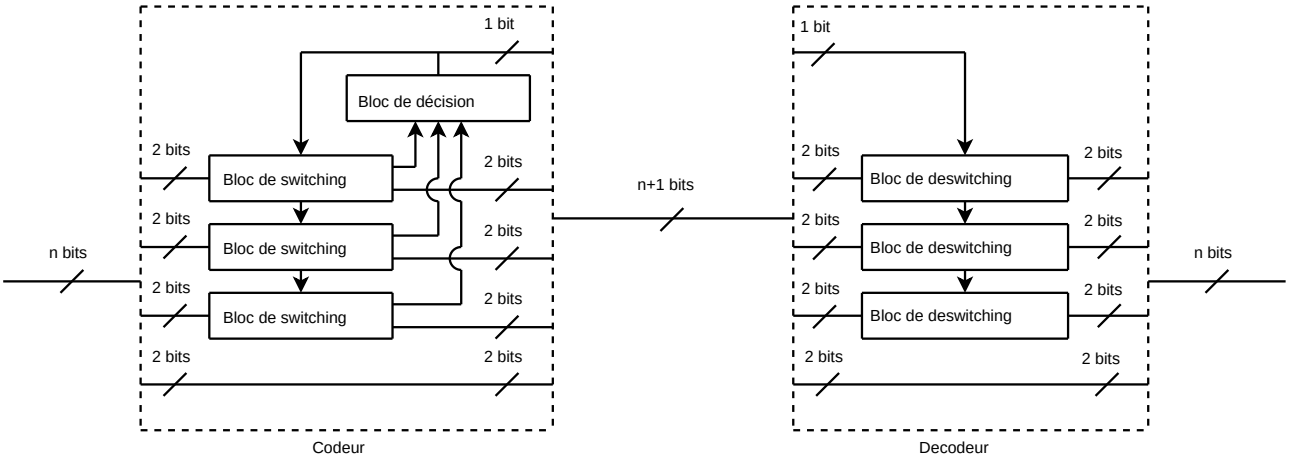


FIG. 4.8 – Architecture des codeur et décodeur utilisés pour le *Spatial Switching* avec un seul fil de contrôle. Exemple pour une application du *Spatial Switching* sur 6 bits.

4.4.1 *Spatial Switching* n'utilisant qu'un seul fil de contrôle

Principe

Si jamais le concepteur est contraint en termes de surface et ne peut se permettre de rajouter un fil de contrôle par bloc de codeur de 2 bits, il est possible d'utiliser un seul fil de contrôle *Switch* pour plusieurs blocs de codeur de 2 bits. Dans ce cas, il est nécessaire de définir un seuil à partir duquel, en fonction du nombre de transitions croisées détectées, le croisement des données sur chaque paire de fils sera fait.
La figure 4.8 illustre ce type d'architecture pour un bus 8 bits sur lequel le *Spatial Switching* n'utilisant qu'un fil de contrôle est appliqué sur les 6 *LSB*.

Résultats expérimentaux

Les expérimentations ont été réalisées pour un bus de 8 bits dans une technologie 90 nm en appliquant le *Spatial Switching* sur les 6 *LSB*. Nous avons testé deux seuils : le croisement des données est déclenché sur les trois paires de fils dès lors qu'une transition croisée dans n'importe laquelle des trois paires de fils est détectée, ou le croisement des données est déclenché sur les trois paires de fils dès lors qu'au moins deux transitions croisées sont détectées. Les résultats expérimentaux présentés à la figure 4.9 représentent trois courbes de gain en consommation sur le bus (incluant la consommation dûe aux codecs) :

- le gain en consommation sur le bus pour le seuil à 1 transition croisée ;
- le gain en consommation sur le bus pour le seuil à 2 transitions croisées ;
- et le gain en consommation du *Spatial Switching* normal (utilisant trois fils de contrôle) à titre de comparaison.

Les résultats expérimentaux montrent qu'il est possible d'obtenir un gain en consommation toujours conséquent. Ce gain sera effectivement moindre qu'en utilisant le *Spatial Switching* normal. En effet, avec un seul fil de contrôle pour les différents blocs, dans certains cas, des transitions croisées ne seront pas supprimées (dans notre exemple, si le seuil vaut 2, tant que l'on n'a qu'une

seule transition croisée sur n'importe quelle paire de fils elle ne sera pas supprimée). Dans d'autres cas, des transitions croisées initialement non-existantes peuvent apparaître (dans notre exemple, si le seuil vaut 1, les trois paires de fils sont croisées si une seule transition croisée a lieu sur n'importe quelle paire de fils. Par conséquent, des transitions sur les paires de fils où l'état des données ne changeait pas peuvent être introduites).

4.4.2 *Spatial Switching* et réorganisation des fils

Principe

Afin de minimiser la consommation des interconnexions, nous pouvons tenter de réduire l'activité des lignes voisines afin de diminuer les transitions croisées. Avant d'appliquer une quelconque technique de codage, regardons l'effet sur la consommation d'un bus en réorganisant l'arrangement des lignes en fonction de leur activité. Nous savons que sur deux lignes voisines qui ont une activité importante, la probabilité d'apparition de transitions croisées sera plus forte. Ici, l'idée est de placer par exemple une ligne avec une activité élevée entre deux lignes dont l'activité est plus faible (i.e. faire une sorte de *shielding* statique dont les fils de blindage sont en fait les fils présentant une activité faible).
La figure 4.10 présente la modification de consommation en réorganisant de différentes manières les fils d'un bus de 8 bis (A_7 étant le bit de poids fort et A_0 celui de poids faible). Sur cette figure le cas a représente la transmission normale du flot de données sans réorganisation des fils. Le cas b représente le déplacement du bit de poids faible A_0 (bit ayant l'activité la plus forte) entre les bits A_6 et A_7 (bits de poids fort ayant les activités les plus faibles). Le cas c représente le déplacement des bits de poids faibles A_0 et A_1 entre les bits de poids fort A_5, A_6 et A_7. De la même manière dans le cas d ce sont les bits de poids faibles A_0, A_1 et A_2 qui sont déplacés. Le maximum de réduction de consommation est obtenu pour ce dernier cas où la réduction de consommation atteind 1.73 %.
Nous allons maintenant combiner le déplacement des bits ayant l'activité la plus importante avec la technique du *Spatial Switching*.

Résultats expérimentaux

Lorsque le *Spatial Switching* est utilisé sur une paire de fils, toutes les transitions croisées (transitions les plus consommatrices) sur cette paire sont éliminées, il n'est donc pas judicieux de déplacer ces fils au risque de créer d'autres transitions croisées avec leurs nouveaux voisins. La réorganisation des fils portera donc sur les fils non codés par le *Spatial Switching*. Nous avons montré dans les résultats expérimentaux sur le *Spatial Switching*, que le maximum de gain sur le bus utilisé lors des expérimentations (i.e. bus de 8 bits) est obtenu pour une utilisation de la méthode sur les 6 *LSB* pour le transfert d'un flot donnée de type image. Nous présenterons donc ici les résultats pour la combinaison du *Spatial Switching* et de la réorganisation dans les mêmes conditions de configuration. Dans cette configuration (codecs sur 6 bits donc 3 fils supplémentaires) le bus est composé de 11 fils. Comme la réorganisation ne porte pas sur les fils codés il reste 5 fils à réorganiser (i.e. les 2 fils de poids fort du bus plus les 3 fils de contrôle issus des blocs de codage).

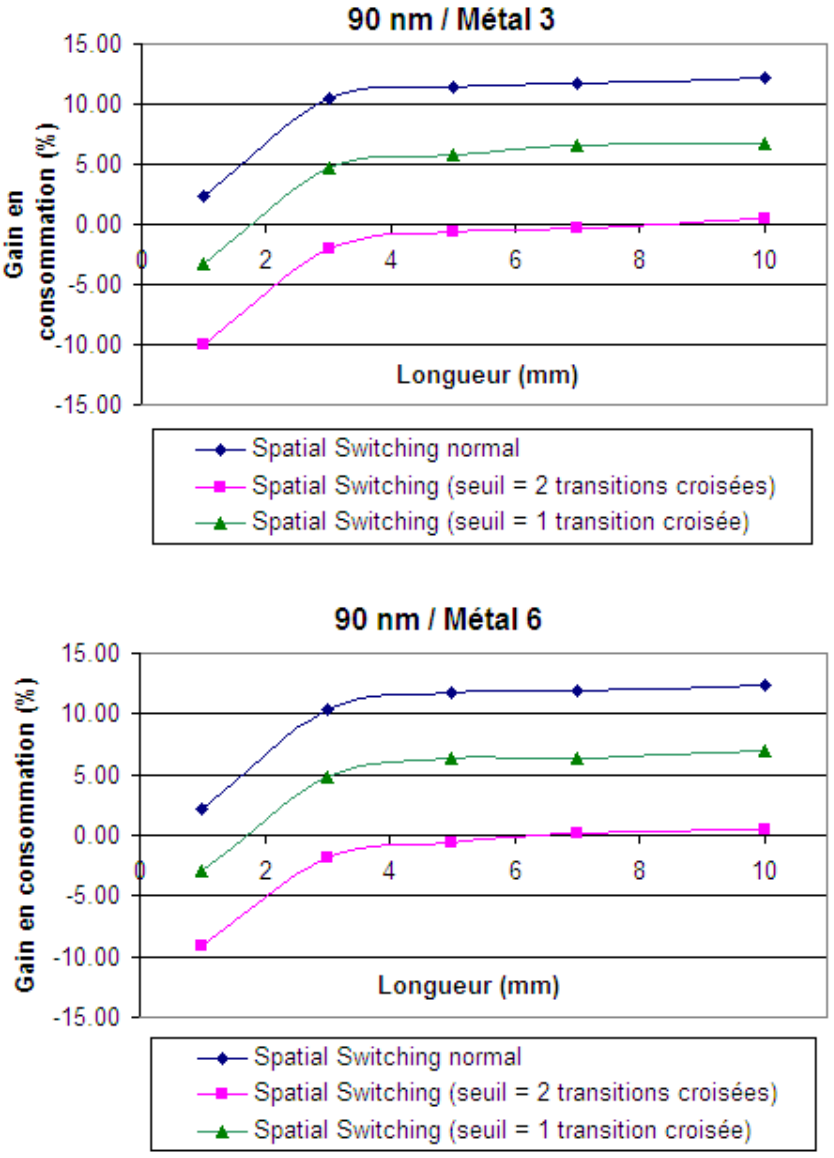


FIG. 4.9 – Comparaison des gains en consommation sur le bus pour l'utilisation du *Spatial Switching* normal et du *Spatial Switching* n'utilisant qu'un fil de contrôle avec un seuil de détection à 1 puis à 2.

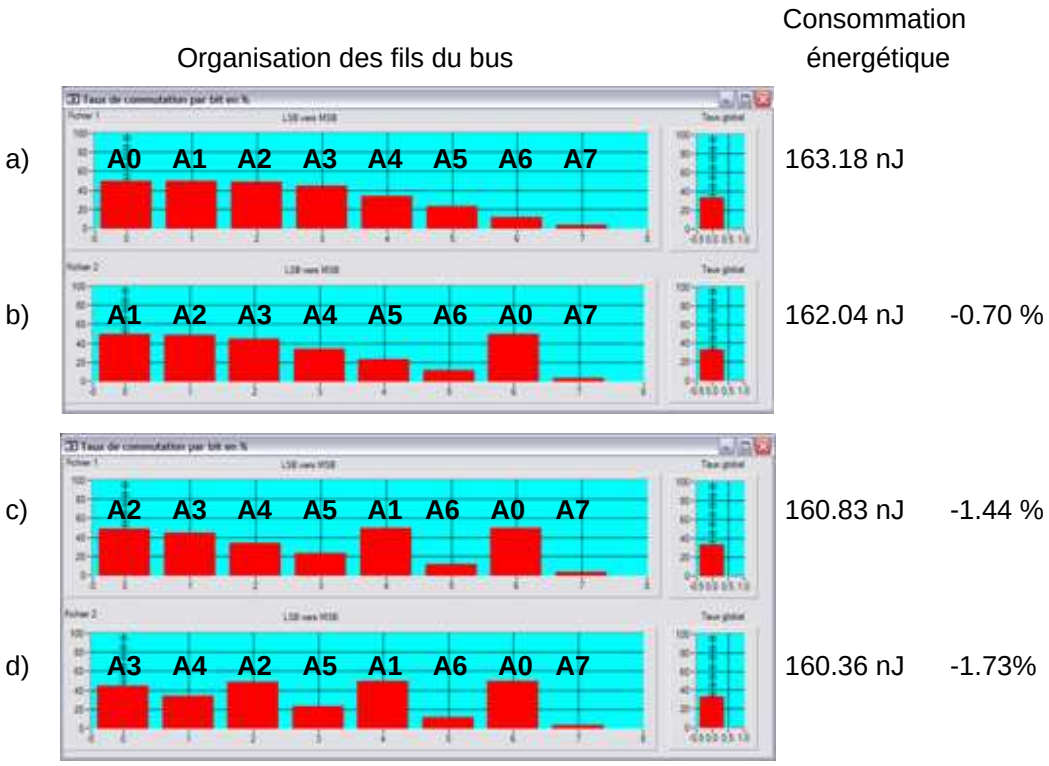


FIG. 4.10 – Comparaison des gains en consommation sur le bus en réorganisant les fils du bus selon l'activité.

Les résultats présentés dans le tableau 4.8 montrent le gain en consommation pour les deux types de configuration (i.e. *Spatial Switching* seul et *Spatial Switching* avec réorganisation). Ces résultats montrent qu'il est possible de gagner encore en consommation (2.15% supplémentaire pour un fil de 3mm par exemple) en positionnant judicieusement les lignes en fonction de leur taux d'activité. Le pourcentage de transitions croisées supprimée est ici de 79%. Si nous comparons ces résultats avec ceux présentés dans la figure 4.7, nous voyons que le pourcentage de transition croisées supprimées augmente de 13% pour les mêmes conditions d'expérience. Donc la réorganisation des fils du bus a un impact non négligeable sur le nombre de transitions croisées pouvant être éliminé, et donc sur la consommation du réseau d'interconnexions.

	Gain en consommation	Gain en consommation
Longueur	*Spatial Switching*	*Spatial Switching* et réorganisation
1 mm	2.39 %	4.52 %
3 mm	10.44 %	12.59 %
5 mm	11.44 %	13.43 %
7 mm	11.72 %	13.79 %
10 mm	12.23 %	13.84 %

TAB. 4.8 – Gain en consommation sur le bus pour le *Spatial Switching* avec réorganisation.

4.5 Bilan

Nous avons présenté dans ce chapitre plusieurs techniques d'optimisation au niveau architectural. Ces techniques, en adéquation avec les critères d'optimisation introduits au chapitre précédent, permettent d'obtenir des gains en consommation intéressants du fait de la simplicité de leur codecs. De plus, contrairement à certaines techniques de la littérature, celles que nous avons proposées ont une efficacité immédiate pour de faible longueurs de bus. De plus, ces gains augmentent avec la longueur de ces derniers et avec les sauts technologiques.

Conclusions et perspectives

Conclusions

Nos travaux de recherche visent à aboutir à une connaissance approfondie de la problématique des interconnexions dans les *SoC* actuels. Les problèmes liés aux interconnexions tels que la fiabilité (bruit dû au *crosstalk*), la consommation et les performances temporelles ont été abordés.
Nous avons vu, dans un premier chapitre, que la modélisation des interconnexions est un phénomène complexe si l'on souhaite obtenir des résultats précis et cohérents lors de l'évaluation du comportement du réseau d'interconnexions. En effet, afin d'obtenir des résultats précis et fiables en termes de délai et de consommation, il est nécessaire de modéliser au niveau physique ces lignes d'interconnexions. Nous avons proposé une modélisation physique résistive et capacitive d'une ligne ainsi que des buffers entrant dans la composition du bus en fonction de la variation des multiples paramètres technologiques. Du fait de la réduction des dimensions technologiques des interconnexions, nous avons également pris en compte, dans notre modélisation, le phénomène de diaphonie capacitive entre les fils. Les enjeux de cette diaphonie (*crosstalk*) en termes de bruit, de délai et de consommation ont également été présentés. Dans le second chapitre, après avoir défini la modélisation physique complète d'un bus et extrait les paramètres importants influants sur le délai et la consommation, une phase de simulation des circuits à l'aide du simulateur ELDO (simulateur SPICE) a été réalisée. A la suite des expérimentations, nous avons conçu un outil d'estimation (*Interconnect Explorer*) permettant à l'utilisateur, après une phase de configuration, d'obtenir rapidement des résultats sur les performances de son réseau d'interconnexions. Nous avons montré que notre outil permet d'obtenir des résultats avec une erreur maximale de 3% (par rapport aux simulations SPICE) avec un temps d'exécution de l'ordre de la seconde (plusieurs heures avec le simulateur SPICE).
La prise en compte du critère de consommation lors de la conception d'un système étant devenu un point stratégique, et tout particulièrement la consommation des interconnexions (plus de 50% de la consommation dynamique d'une puce à l'heure actuelle), nous avons proposé dans le troisième chapitre une analyse bibliographique des techniques de réduction de la consommation et du délai appliquée aux interconnexions. Cette étude bibliographique montre qu'il existe une multitude de techniques d'optimisation applicables aux différents niveaux d'abstraction, les plus nombreuses étant celles proposées au niveau architectural. Ces techniques consistent toutes en un codage des données afin de diminuer leur activité ou de supprimer les transitions engendrant un *crosstalk* important. L'importance du phénomène de *crosstalk* va directement influer sur les performances temporelles et de consommation du réseau. Dans ce chapitre, nous avons également

proposé une analyse des techniques les plus pertinentes à l'aide de notre outil d'estimation, afin d'en extraire l'influence sur la variation des paramètres importants pour la consommation et le délai. Les résultats ont permis de montrer que peu de techniques de la littérature permettent encore d'obtenir des gains en consommation sur les bus *on-chip*, ceci étant principalement dû à la complexité de leur *codecs*. Dans un second temps, cette analyse nous a permis d'extraire de multiples pistes d'optimisation et de proposer un nouveau modèle de consommation.
Nous avons introduit dans le quatrième chapitre plusieurs propositions de techniques d'optimisation au niveau architectural (dont une est brevetée : le *Spatial Switching*) en adéquation avec les pistes extraites précédemment. Elles ont l'avantage de nécessiter des codecs ayant une faible complexité en termes de nombre de transistors. Leur principe étant de supprimer les cas où les transitions présentent un *crosstalk* important, elles ont donc un double effet : réduction de la consommation sur le bus et accélération de la propagation des données engendrés par l'élimination des pires cas de transitions (d'un point de vue temporel et de consommation). Les résultats expérimentaux sur le *Spatial Switching* montrent des gains en consommation pouvant atteindre 12% en incluant évidemment le surcoût de consommation apporté par les codecs. De plus, il est important de souligner que les gains augmentent lors des sauts technologiques. Une comparaison avec la technique du *Partial Bus Invert* a été proposée. Les résultats obtenus montrent que le *Spatial Switching* permet d'obtenir de meilleures performances. Nous avons proposé dans la suite de ce chapitre quelques évolutions possibles de notre méthode dite du *Spatial Switching*.

Perspectives

Nous présentons dans cette section les perspectives découlant de nos travaux. Dans un premier temps, des perspectives à court terme introduiront l'extension de nos travaux par l'élévation du niveau d'abstraction. Enfin, l'utilisation de nos travaux dans les technologies émergentes dans la conception des interconnexions sera présentée dans les perspectives à moyen et long termes.

Perspectives à court terme Les perspectives et les nouveaux axes de recherche à court terme découlant de ces travaux sont relativement nombreux.

Dans un premier temps, une évaluation plus fine de la précision de l'outil peut être effectuée. En effet, il serait intéressant de confronter les résultats d'estimation fournis par *Interconnect Explorer* avec des données extraites d'un layout réel.

Dans un second temps, une analyse des techniques d'optimisation proposées dans ce mémoire en termes de réduction du bruit (ou amélioration du taux d'erreur binaire) peut être proposée. En effet, le fait de réduire (ou d'éliminer) certaines transitions génératrices de bruit (comme présenté dans le premier chapitre) permet de réduire le bruit global et par conséquent d'améliorer le taux d'erreur binaire. L'objectif sera donc de quantifier cette amélioration.

Troisièmement, il est possible, de reprendre les pistes d'optimisation pour la réduction de la consommation afin d'élaborer d'autres techniques de codage des données au niveau architectural.

Approche *Model Driven Engineering* Quatrièmement, nos résultats seront étendus afin d'être utilisés dans une approche *MDE* (*Model Driven Engineering*). Dans ce cadre, nos travaux s'intégreront dans le projet *ITEA SPICES* qui utilise un profil *AADL* (*Application & Architecture Design Language*). Le but d'un profil *AADL* est de faire une description d'un logiciel associé à une plateforme d'exécution par une description à base de composants. Ces composants peuvent appartenir à différentes catégories : catégorie logicielle (*process*, *thread* ...), catégorie composite (*system*) ou catégorie plateforme (processeur, memoire, *device* et bus).
C'est dans cette dernière catégorie, qui contient le composant bus, que peuvent s'insérer nos travaux. En effet, chaque composant dispose de ses propres caractéristiques. Le composant bus peut alors avoir pour caractéristiques celles que nous avons définies comme étant les paramètres d'entrée d'*Interconnect Explorer* ; à savoir la technologie, la couche de métal, la longueur du bus, sa largeur (nombre de bits), sa fréquence de fonctionnement, le type de bufferisation ainsi que les données à estimer. Si le concepteur ne connaît pas certaines de ces valeurs, nous pouvons lui proposer d'effectuer les simulations avec des valeurs prédéterminées. Si par exemple il ne connaît pas les données qui doivent circuler sur son bus, plusieurs profils d'activité lui sont alors proposés. Afin d'estimer la consommation de son bus dès les premières phases de conception, il est possible que le consepteur ne connaisse pas tous les paramètres technologiques de sa cible. Par exemple,

dans le cas où il ne connaît pas la couche de métal sur laquelle son bus sera implémenté, nous pouvons lui proposer d'effectuer les simulations sur plusieurs couches de métal afin de pouvoir borner (borne minimum et maximum) les résultats. De la même manière, s'il ne dispose pas de la longueur du bus, une longueur typique (utilisée dans les *SoC* actuels) avec une plage de variation peut être envisagée pour les simulations. Un raffinement des valeurs des paramètres va naturellement s'opérer au fur et à mesure de la conception et les résultats d'estimation seront donc de plus en plus précis.
Pour cette approche, le cœur de calcul de notre outil s'intégrera sous la forme d'un *plugin* dans le "*framework*" *OSATE* (*Open Source AADL Tool Environment*) afin de pouvoir estimer la consommation des communications dès les premières phases de conception. Cette première approche est en cours de réalisation.

Approche *MPSoC* Enfin, dans ce mémoire, nous nous sommes préoccupés des interconnexions point à point. Or, les systèmes actuels peuvent utiliser des réseaux de communication plus complexes. Par conséquent, il est donc possible de montrer comment utiliser notre approche pour modéliser des interconnexions de type *MESH* ou *NoC* souvent utilisées dans le cadre de systèmes *MPSoC*. Pour cela, nous utiliserons des résultats de la plate forme *SocLib* sur la simulation d'architectures *MPSoC*. Dans cette approche, l'étude portera plus particulièrement sur la modélisation du comportement de la logique de routage (*crossbar*, routeur ...). Il faudra chercher à identifier la part de la consommation de cette logique de routage dans le budget de consommation global du système de communication.
Dans le cas où la consommation de la logique de routage reste minime devant celle des interconnexions, nos modèles pourront être utilisés immédiatement.
Dans le cas contraire, il faudra proposer un modèle de consommation dédié à cette logique. Un modèle suivant le profil d'activité des données peut être envisagé. En effet, les données transitant sur les canaux de communication entre différents émetteurs et récepteurs peuvent être multiplexées temporellement. Il n'est alors plus évident que le profil d'activité des données échangées entre deux entités reste le même tout au long de la propagation sur le réseau d'interconnexions (comparativement à des interconnexions point à point) si plusieurs entités s'échangent des données dans la même fenêtre temporelle. Des modèles de consommation de la logique de routage selon l'évolution de la propagation du profil d'activité en fonction du nombre d'entrées pourraient alors être envisagés.

Perspectives à moyen et long termes Nos travaux (outil et modèles) peuvent être étendus aux solutions alternatives à la conception classique, le but étant ici de reprendre notre méthode de modélisation et de l'appliquer aux technologies emergentes dans la conception d'interconnexions. Trois types de conception nouvelles sont actuellement proposées :

- les interconnections 3D ;
- les interconnexions à base de nanotubes de carbone ;
- et les interconnexions optiques *on-chip*.

Interconnexions 3D En utilisant un système à base d'interconnexions 3D, le routage n'est plus seulement dans le plan comme pour un circuit classique mais se fait également verticalement. Le but d'un routage vertical est d'empiler les uns au dessus des autres les éléments du circuit qui ont besoin de communiquer fréquemment entre eux. Il est possible, par exemple, de mettre un processeur avec sa mémoire cache l'un au dessus de l'autre. De cette manière la longueur du routage est beaucoup plus faible que dans le cas d'une conception à plat. Cette façon de router permet alors une réduction des longueurs et donc une diminution de la consommation et du temps de propagation [DWM+05].
En revanche, puisque les éléments du circuit se situent les uns au dessus des autres, cela pose un problème de dissipation thermique puisque la puissance dissipée doit alors traverser les différentes couches du circuit.
En ce qui concerne la modélisation des interconnexions 3D, la méthode de caractérisation peut être reprise intégralement, puisque le principe du routage des interconnexions reste le même. A la place de réaliser le routage dans le plan, celui-ci est réalisé verticalement. Il suffit alors de disposer du *Design Kit* du constructeur dans lequel apparaissent les notions de dimensions physiques des interconnexions.

Interconnexions à base de nanotubes de carbone Les avancées récentes dans les procédés de fabrication ont permis l'élaboration d'un nouveau type d'interconnexions à base de nanotubes de carbone. Ce type de matériau possède une meilleure conductivité que le cuivre (propagation temporelle plus rapide) ainsi qu'une meilleure conductivité thermique (meilleure dissipation de la chaleur) [Gok05]. Il est également moins sensible au phénomène d'électromigration ce qui permet d'obtenir une meilleure fiabilité dans le temps. Une diminution des capacités parasites est, de plus, observée et plus particulièrement la diminution des capacités latérales (*crosstalk*), ce qui permet d'obtenir de meilleurs résultats en consommation et en délai que la conception classique [NM05].
Par contre, le problème majeur se situe au niveau des règles de dessin. Ce type d'interconnexions ne peut aller qu'en ligne droite ce qui fait que le routage devra également inclure des interconnexions métalliques pour un changement de direction de l'interconnexion [Lie03]. Il est de plus difficile de prévoir les paramètres technologiques résistifs et capacitifs de ce type d'interconnexions puisque la résistance, par exemple, ne varie pas de manière linéaire avec la longueur de la ligne.
De la même manière que précédemment, la méthode de caractérisation peut être utilisée pour les interconnexions à base de nanotubes de carbone. Le seul point qui peut être gênant à l'heure actuelle, est la difficulté de prévoir les paramètres résistifs et capacitifs de ce type de technologie pour la modélisation.

Interconnexions optiques *on-chip* Une dernière alternative à la conception classique proposée sont les interconnexions optiques [CCH+06]. Ce type d'interconnexions est utilisé pour transmettre des signaux sur de longues distances à l'intérieur d'une puce. Le gros avantage, en ce qui concerne les bus, est qu'il est possible avec cette technologie de transmettre sur un même conducteur optique plusieurs signaux en multiplexant les différentes longueurs d'ondes [Kob05].

Il n'y a ici plus aucun phénomène de *crosstalk*. De plus, pour la transmission des signaux *on-chip*, aucun répéteur n'est nécessaire par rapport à la conception classique.
En revanche, il est nécessaire de disposer de convertisseurs électrique vers optique et optique vers électrique au niveau de l'émetteur et respectivement du récepteur. Malheureusement, ces convertisseurs sont à l'heure actuelle complexes et gourmands en surface occupée sur la puce. De plus, afin de justifier l'utilisation de la solution optique, il faut que le coût en consommation du système optique soit inférieur à celui de la solution conventionnelle.
Pour ce dernier point, nos travaux peuvent être utilisés à titre de comparaison, afin de justifier le choix d'un changement de procédé de communication si la consommation de ce dernier est inférieure à celle fournie par notre estimateur.

Annexes

Grandeur	Unité	Symbole
Résistance	Ohm	Ω
Résistance par unité de longueur	Ohm/mètre	Ω/m
Capacité	Farad	F
Capacité par unité de longueur	Farad/mètre	F/m
Inductance	Henry	H
Résistivité	Ohm.mètre	$\Omega.m$
Permitivité	Farad/mètre	F/m
Mobilité	mètre carré/Volt.seconde	$m^2/V.s$
Dimension	Mètre	m
Tension	Volt	V
Surface	mètre carré	m^2
Fréquence	Hertz	Hz
Temps	Seconde	s
Energie	Joule	J
Puissance	Watt	W

TAB. 4.9 – Définition des grandeurs, de leur unités et de leur symboles.

```
* PTM 130nm NMOS

.model  nmos  nmos  level = 54

+version = 4.0          binunit = 1           paramchk= 1           mobmod  = 0
+capmod  = 2            igcmod  = 1           igbmod  = 1           geomod  = 1
+diomod  = 1            rdsmod  = 0           rbodymod= 1           rgatemod= 1
+permod  = 1            acnqsmod= 0           trnqsmod= 0

+tnom    = 27           toxe    = 2.25e-9     toxp    = 1.6e-9      toxm    = 2.25e-9
+dtox    = 0.65e-9      epsrox  = 3.9         wint    = 5e-009      lint    = 10.5e-009
+ll      = 0            wl      = 0           lln     = 1           wln     = 1
+lw      = 0            ww      = 0           lwn     = 1           wwn     = 1
+lwl     = 0            wwl     = 0           xpart   = 0           toxref  = 2.25e-9
+xl      = -60e-9
+vth0    = 0.3782       k1      = 0.4         k2      = 0.01        k3      = 0
+k3b     = 0            w0      = 2.5e-006    dvt0    = 1           dvt1    = 2
+dvt2    = -0.032       dvt0w   = 0           dvt1w   = 0           dvt2w   = 0
+dsub    = 0.1          minv    = 0.05        voffl   = 0           dvtp0   = 1.2e-010
+dvtp1   = 0.1          lpe0    = 0           lpeb    = 0           xj      = 3.92e-008
+ngate   = 2e+020       ndep    = 1.54e+018   nsd     = 2e+020      phin    = 0
+cdsc    = 0.0002       cdscb   = 0           cdscd   = 0           cit     = 0
+voff    = -0.13        nfactor = 1.5         eta0    = 0.0092      etab    = 0
```

```
+vfb     = -0.55        u0      = 0.05928      ua      = 6e-010       ub      = 1.2e-018
+uc      = 0            vsat    = 100370       a0      = 1            ags     = 1e-020
+a1      = 0            a2      = 1            b0      = 0            b1      = 0
+keta    = 0.04         dwg     = 0            dwb     = 0            pclm    = 0.06
+pdiblc1 = 0.001        pdiblc2 = 0.001        pdiblcb = -0.005       drout   = 0.5
+pvag    = 1e-020       delta   = 0.01         pscbe1  = 8.14e+008    pscbe2  = 1e-007
+fprout  = 0.2          pdits   = 0.08         pditsd  = 0.23         pditsl  = 2.3e+006
+rsh     = 5            rdsw    = 200          rsw     = 100          rdw     = 100
+rdswmin = 0            rdwmin  = 0            rswmin  = 0            prwg    = 0
+prwb    = 6.8e-011     wr      = 1            alpha0  = 0.074        alpha1  = 0.005
+beta0   = 30           agidl   = 0.0002       bgidl   = 2.1e+009     cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012        bigbacc = 0.0028       cigbacc = 0.002
+nigbacc = 1            aigbinv = 0.014        bigbinv = 0.004        cigbinv = 0.004
+eigbinv = 1.1          nigbinv = 3            aigc    = 0.012        bigc    = 0.0028
+cigc    = 0.002        aigsd   = 0.012        bigsd   = 0.0028       cigsd   = 0.002
+nigc    = 1            poxedge = 1            pigcd   = 1            ntox    = 1

+xrcrg1  = 12           xrcrg2  = 5
+cgso    = 2.4e-010     cgdo    = 2.4e-010     cgbo    = 2.56e-011    cgdl    = 2.653e-10
+cgsl    = 2.653e-10    ckappas = 0.03         ckappad = 0.03         acde    = 1
+moin    = 15           noff    = 0.9          voffcv  = 0.02

+kt1     = -0.11        kt1l    = 0            kt2     = 0.022        ute     = -1.5
+ua1     = 4.31e-009    ub1     = 7.61e-018    uc1     = -5.6e-011    prt     = 0
+at      = 33000

+fnoimod = 1            tnoimod = 0

+jss     = 0.0001       jsws    = 1e-011       jswgs   = 1e-010       njs     = 1
+ijthsfwd= 0.01         ijthsrev= 0.001        bvs     = 10           xjbvs   = 1
+jsd     = 0.0001       jswd    = 1e-011       jswgd   = 1e-010       njd     = 1
+ijthdfwd= 0.01         ijthdrev= 0.001        bvd     = 10           xjbvd   = 1
+pbs     = 1            cjs     = 0.0005       mjs     = 0.5          pbsws   = 1
+cjsws   = 5e-010       mjsws   = 0.33         pbswgs  = 1            cjswgs  = 3e-010
+mjswgs  = 0.33         pbd     = 1            cjd     = 0.0005       mjd     = 0.5
+pbswd   = 1            cjswd   = 5e-010       mjswd   = 0.33         pbswgd  = 1
+cjswgd  = 5e-010       mjswgd  = 0.33         tpb     = 0.005        tcj     = 0.001
+tpbsw   = 0.005        tcjsw   = 0.001        tpbswg  = 0.005        tcjswg  = 0.001
+xtis    = 3            xtid    = 3

+dmcg    = 0e-006       dmci    = 0e-006       dmdg    = 0e-006       dmcgt   = 0e-007
+dwj     = 0.0e-008     xgw     = 0e-007       xgl     = 0e-008

+rshg    = 0.4          gbmin   = 1e-010       rbpb    = 5            rbpd    = 15
+rbps    = 15           rbdb    = 15           rbsb    = 15           ngcon   = 1

* PTM 130nm PMOS

.model  pmos  pmos  level = 54

+version = 4.0          binunit = 1            paramchk= 1            mobmod  = 0
+capmod  = 2            igcmod  = 1            igbmod  = 1            geomod  = 1
+diomod  = 1            rdsmod  = 0            rbodymod= 1            rgatemod= 1
+permod  = 1            acnqsmod= 0            trnqsmod= 0

+tnom    = 27           toxe    = 2.35e-009    toxp    = 1.6e-009     toxm    = 2.35e-009
+dtox    = 0.75e-9      epsrox  = 3.9          wint    = 5e-009       lint    = 10.5e-009
+ll      = 0            wl      = 0            lln     = 1            wln     = 1
```

```
+lw      = 0            ww      = 0            lwn     = 1            wwn     = 1
+lwl     = 0            wwl     = 0            xpart   = 0            toxref  = 2.35e-009
+xl      = -60e-9
+vth0    = -0.321       k1      = 0.4          k2      = -0.01        k3      = 0
+k3b     = 0            w0      = 2.5e-006     dvt0    = 1            dvt1    = 2
+dvt2    = -0.032       dvt0w   = 0            dvt1w   = 0            dvt2w   = 0
+dsub    = 0.1          minv    = 0.05         voffl   = 0            dvtp0   = 1e-009
+dvtp1   = 0.05         lpe0    = 0            lpeb    = 0            xj      = 3.92e-008
+ngate   = 2e+020       ndep    = 1.14e+018    nsd     = 2e+020       phin    = 0
+cdsc    = 0.000258     cdscb   = 0            cdscd   = 6.1e-008     cit     = 0
+voff    = -0.126       nfactor = 1.5          eta0    = 0.0092       etab    = 0
+vfb     = 0.55         u0      = 0.00835      ua      = 2.0e-009     ub      = 0.5e-018
+uc      = -3e-011      vsat    = 70000        a0      = 1.0          ags     = 1e-020
+a1      = 0            a2      = 1            b0      = -1e-020      b1      = 0
+keta    = -0.047       dwg     = 0            dwb     = 0            pclm    = 0.12
+pdiblc1 = 0.001        pdiblc2 = 0.001        pdiblcb = 3.4e-008     drout   = 0.56
+pvag    = 1e-020       delta   = 0.01         pscbe1  = 8.14e+008    pscbe2  = 9.58e-007
+fprout  = 0.2          pdits   = 0.08         pditsd  = 0.23         pditsl  = 2.3e+006
+rsh     = 5            rdsw    = 240          rsw     = 120          rdw     = 120
+rdswmin = 0            rdwmin  = 0            rswmin  = 0            prwg    = 3.22e-008
+prwb    = 6.8e-011     wr      = 1            alpha0  = 0.074        alpha1  = 0.005
+beta0   = 30           agidl   = 0.0002       bgidl   = 2.1e+009     cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012        bigbacc = 0.0028       cigbacc = 0.002
+nigbacc = 1            aigbinv = 0.014        bigbinv = 0.004        cigbinv = 0.004
+eigbinv = 1.1          nigbinv = 3            aigc    = 0.69         bigc    = 0.0012
+cigc    = 0.0008       aigsd   = 0.0087       bigsd   = 0.0012       cigsd   = 0.0008
+nigc    = 1            poxedge = 1            pigcd   = 1            ntox    = 1

+xrcrg1  = 12           xrcrg2  = 5
+cgso    = 2.4e-010     cgdo    = 2.4e-010     cgbo    = 2.56e-011    cgdl    = 2.653e-10
+cgsl    = 2.653e-10    ckappas = 0.03         ckappad = 0.03         acde    = 1
+moin    = 15           noff    = 0.9          voffcv  = 0.02

+kt1     = -0.11        kt1l    = 0            kt2     = 0.022        ute     = -1.5
+ua1     = 4.31e-009    ub1     = 7.61e-018    uc1     = -5.6e-011    prt     = 0
+at      = 33000

+fnoimod = 1            tnoimod = 0

+jss     = 0.0001       jsws    = 1e-011       jswgs   = 1e-010       njs     = 1
+ijthsfwd= 0.01         ijthsrev= 0.001        bvs     = 10           xjbvs   = 1
+jsd     = 0.0001       jswd    = 1e-011       jswgd   = 1e-010       njd     = 1
+ijthdfwd= 0.01         ijthdrev= 0.001        bvd     = 10           xjbvd   = 1
+pbs     = 1            cjs     = 0.0005       mjs     = 0.5          pbsws   = 1
+cjsws   = 5e-010       mjsws   = 0.33         pbswgs  = 1            cjswgs  = 3e-010
+mjswgs  = 0.33         pbd     = 1            cjd     = 0.0005       mjd     = 0.5
+pbswd   = 1            cjswd   = 5e-010       mjswd   = 0.33         pbswgd  = 1
+cjswgd  = 5e-010       mjswgd  = 0.33         tpb     = 0.005        tcj     = 0.001
+tpbsw   = 0.005        tcjsw   = 0.001        tpbswg  = 0.005        tcjswg  = 0.001
+xtis    = 3            xtid    = 3

+dmcg    = 0e-006       dmci    = 0e-006       dmdg    = 0e-006       dmcgt   = 0e-007
+dwj     = 0.0e-008     xgw     = 0e-007       xgl     = 0e-008

+rshg    = 0.4          gbmin   = 1e-010       rbpb    = 5            rbpd    = 15
+rbps    = 15           rbdb    = 15           rbsb    = 15           ngcon   = 1
```

```
* PTM 90nm NMOS

.model  nmos  nmos  level = 54

+version = 4.0          binunit = 1           paramchk= 1           mobmod  = 0
+capmod  = 2            igcmod  = 1           igbmod  = 1           geomod  = 1
+diomod  = 1            rdsmod  = 0           rbodymod= 1           rgatemod= 1
+permod  = 1            acnqsmod= 0           trnqsmod= 0

+tnom    = 27           toxe    = 2.05e-9     toxp    = 1.4e-9      toxm    = 2.05e-9
+dtox    = 0.65e-9      epsrox  = 3.9         wint    = 5e-009      lint    = 7.5e-009
+ll      = 0            wl      = 0           lln     = 1           wln     = 1
+lw      = 0            ww      = 0           lwn     = 1           wwn     = 1
+lwl     = 0            wwl     = 0           xpart   = 0           toxref  = 2.05e-9
+xl      = -40e-9
+vth0    = 0.397        k1      = 0.4         k2      = 0.01        k3      = 0
+k3b     = 0            w0      = 2.5e-006    dvt0    = 1           dvt1    = 2
+dvt2    = -0.032       dvt0w   = 0           dvt1w   = 0           dvt2w   = 0
+dsub    = 0.1          minv    = 0.05        voffl   = 0           dvtp0   = 1.2e-009
+dvtp1   = 0.1          lpe0    = 0           lpeb    = 0           xj      = 2.8e-008
+ngate   = 2e+020       ndep    = 1.94e+018   nsd     = 2e+020      phin    = 0
+cdsc    = 0.0002       cdscb   = 0           cdscd   = 0           cit     = 0
+voff    = -0.13        nfactor = 1.7         eta0    = 0.0074      etab    = 0
+vfb     = -0.55        u0      = 0.0547      ua      = 6e-010      ub      = 1.2e-018
+uc      = -3e-011      vsat    = 113760      a0      = 1.0         ags     = 1e-020
+a1      = 0            a2      = 1           b0      = -1e-020     b1      = 0
+keta    = 0.04         dwg     = 0           dwb     = 0           pclm    = 0.06
+pdiblc1 = 0.001        pdiblc2 = 0.001       pdiblcb = -0.005      drout   = 0.5
+pvag    = 1e-020       delta   = 0.01        pscbe1  = 8.14e+008   pscbe2  = 1e-007
+fprout  = 0.2          pdits   = 0.08        pditsd  = 0.23        pditsl  = 2.3e+006
+rsh     = 5            rdsw    = 180         rsw     = 90          rdw     = 90
+rdswmin = 0            rdwmin  = 0           rswmin  = 0           prwg    = 0
+prwb    = 6.8e-011     wr      = 1           alpha0  = 0.074       alpha1  = 0.005
+beta0   = 30           agidl   = 0.0002      bgidl   = 2.1e+009    cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012        bigbacc = 0.0028      cigbacc = 0.002
+nigbacc = 1            aigbinv = 0.014       bigbinv = 0.004       cigbinv = 0.004
+eigbinv = 1.1          nigbinv = 3           aigc    = 0.012       bigc    = 0.0028
+cigc    = 0.002        aigsd   = 0.012       bigsd   = 0.0028      cigsd   = 0.002
+nigc    = 1            poxedge = 1           pigcd   = 1           ntox    = 1

+xrcrg1  = 12           xrcrg2  = 5
+cgso    = 1.9e-010     cgdo    = 1.9e-010    cgbo    = 2.56e-011   cgdl    = 2.653e-10
+cgsl    = 2.653e-10    ckappas = 0.03        ckappad = 0.03        acde    = 1
+moin    = 15           noff    = 0.9         voffcv  = 0.02

+kt1     = -0.11        kt1l    = 0           kt2     = 0.022       ute     = -1.5
+ua1     = 4.31e-009    ub1     = 7.61e-018   uc1     = -5.6e-011   prt     = 0
+at      = 33000

+fnoimod = 1            tnoimod = 0

+jss     = 0.0001       jsws    = 1e-011      jswgs   = 1e-010      njs     = 1
+ijthsfwd= 0.01         ijthsrev= 0.001       bvs     = 10          xjbvs   = 1
+jsd     = 0.0001       jswd    = 1e-011      jswgd   = 1e-010      njd     = 1
+ijthdfwd= 0.01         ijthdrev= 0.001       bvd     = 10          xjbvd   = 1
+pbs     = 1            cjs     = 0.0005      mjs     = 0.5         pbsws   = 1
```

```
+cjsws   = 5e-010       mjsws   = 0.33         pbswgs  = 1            cjswgs  = 3e-010
+mjswgs  = 0.33         pbd     = 1            cjd     = 0.0005       mjd     = 0.5
+pbswd   = 1            cjswd   = 5e-010       mjswd   = 0.33         pbswgd  = 1
+cjswgd  = 5e-010       mjswgd  = 0.33         tpb     = 0.005        tcj     = 0.001
+tpbsw   = 0.005        tcjsw   = 0.001        tpbswg  = 0.005        tcjswg  = 0.001
+xtis    = 3            xtid    = 3

+dmcg    = 0e-006       dmci    = 0e-006       dmdg    = 0e-006       dmcgt   = 0e-007
+dwj     = 0.0e-008     xgw     = 0e-007       xgl     = 0e-008

+rshg    = 0.4          gbmin   = 1e-010       rbpb    = 5            rbpd    = 15
+rbps    = 15           rbdb    = 15           rbsb    = 15           ngcon   = 1

* PTM 90nm PMOS

.model  pmos  pmos  level = 54

+version = 4.0          binunit = 1            paramchk= 1            mobmod  = 0
+capmod  = 2            igcmod  = 1            igbmod  = 1            geomod  = 1
+diomod  = 1            rdsmod  = 0            rbodymod= 1            rgatemod= 1
+permod  = 1            acnqsmod= 0            trnqsmod= 0

+tnom    = 27           toxe    = 2.15e-009    toxp    = 1.4e-009     toxm    = 2.15e-009
+dtox    = 0.75e-9      epsrox  = 3.9          wint    = 5e-009       lint    = 7.5e-009
+ll      = 0            wl      = 0            lln     = 1            wln     = 1
+lw      = 0            ww      = 0            lwn     = 1            wwn     = 1
+lwl     = 0            wwl     = 0            xpart   = 0            toxref  = 2.15e-009
+xl      = -40e-9
+vth0    = -0.339       k1      = 0.4          k2      = -0.01        k3      = 0
+k3b     = 0            w0      = 2.5e-006     dvt0    = 1            dvt1    = 2
+dvt2    = -0.032       dvt0w   = 0            dvt1w   = 0            dvt2w   = 0
+dsub    = 0.1          minv    = 0.05         voffl   = 0            dvtp0   = 1e-009
+dvtp1   = 0.05         lpe0    = 0            lpeb    = 0            xj      = 2.8e-008
+ngate   = 2e+020       ndep    = 1.43e+018    nsd     = 2e+020       phin    = 0
+cdsc    = 0.000258     cdscb   = 0            cdscd   = 6.1e-008     cit     = 0
+voff    = -0.126       nfactor = 1.7          eta0    = 0.0074       etab    = 0
+vfb     = 0.55         u0      = 0.00711      ua      = 2.0e-009     ub      = 0.5e-018
+uc      = -3e-011      vsat    = 70000        a0      = 1.0          ags     = 1e-020
+a1      = 0            a2      = 1            b0      = 0            b1      = 0
+keta    = -0.047       dwg     = 0            dwb     = 0            pclm    = 0.12
+pdiblc1 = 0.001        pdiblc2 = 0.001        pdiblcb = 3.4e-008     drout   = 0.56
+pvag    = 1e-020       delta   = 0.01         pscbe1  = 8.14e+008    pscbe2  = 9.58e-007
+fprout  = 0.2          pdits   = 0.08         pditsd  = 0.23         pditsl  = 2.3e+006
+rsh     = 5            rdsw    = 200          rsw     = 100          rdw     = 100
+rdswmin = 0            rdwmin  = 0            rswmin  = 0            prwg    = 3.22e-008
+prwb    = 6.8e-011     wr      = 1            alpha0  = 0.074        alpha1  = 0.005
+beta0   = 30           agidl   = 0.0002       bgidl   = 2.1e+009     cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012        bigbacc = 0.0028       cigbacc = 0.002
+nigbacc = 1            aigbinv = 0.014        bigbinv = 0.004        cigbinv = 0.004
+eigbinv = 1.1          nigbinv = 3            aigc    = 0.69         bigc    = 0.0012
+cigc    = 0.0008       aigsd   = 0.0087       bigsd   = 0.0012       cigsd   = 0.0008
+nigc    = 1            poxedge = 1            pigcd   = 1            ntox    = 1

+xrcrg1  = 12           xrcrg2  = 5
+cgso    = 1.8e-010     cgdo    = 1.8e-010     cgbo    = 2.56e-011    cgdl    = 2.653e-10
+cgsl    = 2.653e-10    ckappas = 0.03         ckappad = 0.03         acde    = 1
+moin    = 15           noff    = 0.9          voffcv  = 0.02
```

```
+kt1     = -0.11        kt1l    = 0            kt2     = 0.022        ute     = -1.5
+ua1     = 4.31e-009    ub1     = 7.61e-018    uc1     = -5.6e-011    prt     = 0
+at      = 33000

+fnoimod = 1            tnoimod = 0

+jss     = 0.0001       jsws    = 1e-011       jswgs   = 1e-010       njs     = 1
+ijthsfwd= 0.01         ijthsrev= 0.001        bvs     = 10           xjbvs   = 1
+jsd     = 0.0001       jswd    = 1e-011       jswgd   = 1e-010       njd     = 1
+ijthdfwd= 0.01         ijthdrev= 0.001        bvd     = 10           xjbvd   = 1
+pbs     = 1            cjs     = 0.0005       mjs     = 0.5          pbsws   = 1
+cjsws   = 5e-010       mjsws   = 0.33         pbswgs  = 1            cjswgs  = 3e-010
+mjswgs  = 0.33         pbd     = 1            cjd     = 0.0005       mjd     = 0.5
+pbswd   = 1            cjswd   = 5e-010       mjswd   = 0.33         pbswgd  = 1
+cjswgd  = 5e-010       mjswgd  = 0.33         tpb     = 0.005        tcj     = 0.001
+tpbsw   = 0.005        tcjsw   = 0.001        tpbswg  = 0.005        tcjswg  = 0.001
+xtis    = 3            xtid    = 3

+dmcg    = 0e-006       dmci    = 0e-006       dmdg    = 0e-006       dmcgt   = 0e-007
+dwj     = 0.0e-008     xgw     = 0e-007       xgl     = 0e-008

+rshg    = 0.4          gbmin   = 1e-010       rbpb    = 5            rbpd    = 15
+rbps    = 15           rbdb    = 15           rbsb    = 15           ngcon   = 1



* PTM 65nm NMOS

.model  nmos  nmos  level = 54

+version = 4.0          binunit = 1            paramchk= 1            mobmod  = 0
+capmod  = 2            igcmod  = 1            igbmod  = 1            geomod  = 1
+diomod  = 1            rdsmod  = 0            rbodymod= 1            rgatemod= 1
+permod  = 1            acnqsmod= 0            trnqsmod= 0

+tnom    = 27           toxe    = 1.85e-9      toxp    = 1.2e-9       toxm    = 1.85e-9
+dtox    = 0.65e-9      epsrox  = 3.9          wint    = 5e-009       lint    = 5.25e-009
+ll      = 0            wl      = 0            lln     = 1            wln     = 1
+lw      = 0            ww      = 0            lwn     = 1            wwn     = 1
+lwl     = 0            wwl     = 0            xpart   = 0            toxref  = 1.85e-9
+xl      = -30e-9
+vth0    = 0.423        k1      = 0.4          k2      = 0.01         k3      = 0
+k3b     = 0            w0      = 2.5e-006     dvt0    = 1            dvt1    = 2
+dvt2    = -0.032       dvt0w   = 0            dvt1w   = 0            dvt2w   = 0
+dsub    = 0.1          minv    = 0.05         voffl   = 0            dvtp0   = 1.0e-009
+dvtp1   = 0.1          lpe0    = 0            lpeb    = 0            xj      = 1.96e-008
+ngate   = 2e+020       ndep    = 2.54e+018    nsd     = 2e+020       phin    = 0
+cdsc    = 0.000        cdscb   = 0            cdscd   = 0            cit     = 0
+voff    = -0.13        nfactor = 1.9          eta0    = 0.0058       etab    = 0
+vfb     = -0.55        u0      = 0.0491       ua      = 6e-010       ub      = 1.2e-018
+uc      = 0            vsat    = 124340       a0      = 1.0          ags     = 1e-020
+a1      = 0            a2      = 1.0          b0      = 0            b1      = 0
+keta    = 0.04         dwg     = 0            dwb     = 0            pclm    = 0.04
+pdiblc1 = 0.001        pdiblc2 = 0.001        pdiblcb = -0.005       drout   = 0.5
+pvag    = 1e-020       delta   = 0.01         pscbe1  = 8.14e+008    pscbe2  = 1e-007
+fprout  = 0.2          pdits   = 0.08         pditsd  = 0.23         pditsl  = 2.3e+006
+rsh     = 5            rdsw    = 165          rsw     = 85           rdw     = 85
+rdswmin = 0            rdwmin  = 0            rswmin  = 0            prwg    = 0
+prwb    = 6.8e-011     wr      = 1            alpha0  = 0.074        alpha1  = 0.005
```

```
+beta0   = 30          agidl   = 0.0002      bgidl   = 2.1e+009    cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012       bigbacc = 0.0028      cigbacc = 0.002
+nigbacc = 1           aigbinv = 0.014       bigbinv = 0.004       cigbinv = 0.004
+eigbinv = 1.1         nigbinv = 3           aigc    = 0.012       bigc    = 0.0028
+cigc    = 0.002       aigsd   = 0.012       bigsd   = 0.0028      cigsd   = 0.002
+nigc    = 1           poxedge = 1           pigcd   = 1           ntox    = 1

+xrcrg1  = 12          xrcrg2  = 5
+cgso    = 1.5e-010    cgdo    = 1.5e-010    cgbo    = 2.56e-011   cgdl    = 2.653e-10
+cgsl    = 2.653e-10   ckappas = 0.03        ckappad = 0.03        acde    = 1
+moin    = 15          noff    = 0.9         voffcv  = 0.02

+kt1     = -0.11       kt1l    = 0           kt2     = 0.022       ute     = -1.5
+ua1     = 4.31e-009   ub1     = 7.61e-018   uc1     = -5.6e-011   prt     = 0
+at      = 33000

+fnoimod = 1           tnoimod = 0

+jss     = 0.0001      jsws    = 1e-011      jswgs   = 1e-010      njs     = 1
+ijthsfwd= 0.01        ijthsrev= 0.001       bvs     = 10          xjbvs   = 1
+jsd     = 0.0001      jswd    = 1e-011      jswgd   = 1e-010      njd     = 1
+ijthdfwd= 0.01        ijthdrev= 0.001       bvd     = 10          xjbvd   = 1
+pbs     = 1           cjs     = 0.0005      mjs     = 0.5         pbsws   = 1
+cjsws   = 5e-010      mjsws   = 0.33        pbswgs  = 1           cjswgs  = 3e-010
+mjswgs  = 0.33        pbd     = 1           cjd     = 0.0005      mjd     = 0.5
+pbswd   = 1           cjswd   = 5e-010      mjswd   = 0.33        pbswgd  = 1
+cjswgd  = 5e-010      mjswgd  = 0.33        tpb     = 0.005       tcj     = 0.001
+tpbsw   = 0.005       tcjsw   = 0.001       tpbswg  = 0.005       tcjswg  = 0.001
+xtis    = 3           xtid    = 3

+dmcg    = 0e-006      dmci    = 0e-006      dmdg    = 0e-006      dmcgt   = 0e-007
+dwj     = 0.0e-008    xgw     = 0e-007      xgl     = 0e-008

+rshg    = 0.4         gbmin   = 1e-010      rbpb    = 5           rbpd    = 15
+rbps    = 15          rbdb    = 15          rbsb    = 15          ngcon   = 1

* PTM 65nm PMOS

.model  pmos  pmos  level = 54

+version = 4.0         binunit = 1           paramchk= 1           mobmod  = 0
+capmod  = 2           igcmod  = 1           igbmod  = 1           geomod  = 1
+diomod  = 1           rdsmod  = 0           rbodymod= 1           rgatemod= 1
+permod  = 1           acnqsmod= 0           trnqsmod= 0

+tnom    = 27          toxe    = 1.95e-009   toxp    = 1.2e-009    toxm    = 1.95e-009
+dtox    = 0.75e-9     epsrox  = 3.9         wint    = 5e-009      lint    = 5.25e-009
+ll      = 0           wl      = 0           lln     = 1           wln     = 1
+lw      = 0           ww      = 0           lwn     = 1           wwn     = 1
+lwl     = 0           wwl     = 0           xpart   = 0           toxref  = 1.95e-009
+xl      = -30e-9
+vth0    = -0.365      k1      = 0.4         k2      = -0.01       k3      = 0
+k3b     = 0           w0      = 2.5e-006    dvt0    = 1           dvt1    = 2
+dvt2    = -0.032      dvt0w   = 0           dvt1w   = 0           dvt2w   = 0
+dsub    = 0.1         minv    = 0.05        voffl   = 0           dvtp0   = 1e-009
+dvtp1   = 0.05        lpe0    = 0           lpeb    = 0           xj      = 1.96e-008
+ngate   = 2e+020      ndep    = 1.87e+018   nsd     = 2e+020      phin    = 0
+cdsc    = 0.000       cdscb   = 0           cdscd   = 0           cit     = 0
```

```
+voff    = -0.126      nfactor = 1.9         eta0    = 0.0058      etab    = 0
+vfb     = 0.55        u0      = 0.00574     ua      = 2.0e-009    ub      = 0.5e-018
+uc      = 0           vsat    = 70000       a0      = 1.0         ags     = 1e-020
+a1      = 0           a2      = 1           b0      = -1e-020     b1      = 0
+keta    = -0.047      dwg     = 0           dwb     = 0           pclm    = 0.12
+pdiblc1 = 0.001       pdiblc2 = 0.001       pdiblcb = 3.4e-008    drout   = 0.56
+pvag    = 1e-020      delta   = 0.01        pscbe1  = 8.14e+008   pscbe2  = 9.58e-007
+fprout  = 0.2         pdits   = 0.08        pditsd  = 0.23        pditsl  = 2.3e+006
+rsh     = 5           rdsw    = 165         rsw     = 85          rdw     = 85
+rdswmin = 0           rdwmin  = 0           rswmin  = 0           prwg    = 3.22e-008
+prwb    = 6.8e-011    wr      = 1           alpha0  = 0.074       alpha1  = 0.005
+beta0   = 30          agidl   = 0.0002      bgidl   = 2.1e+009    cgidl   = 0.0002
+egidl   = 0.8

+aigbacc = 0.012       bigbacc = 0.0028      cigbacc = 0.002
+nigbacc = 1           aigbinv = 0.014       bigbinv = 0.004       cigbinv = 0.004
+eigbinv = 1.1         nigbinv = 3           aigc    = 0.69        bigc    = 0.0012
+cigc    = 0.0008      aigsd   = 0.0087      bigsd   = 0.0012      cigsd   = 0.0008
+nigc    = 1           poxedge = 1           pigcd   = 1           ntox    = 1

+xrcrg1  = 12          xrcrg2  = 5
+cgso    = 1.5e-010    cgdo    = 1.5e-010    cgbo    = 2.56e-011   cgdl    = 2.653e-10
+cgsl    = 2.653e-10   ckappas = 0.03        ckappad = 0.03        acde    = 1
+moin    = 15          noff    = 0.9         voffcv  = 0.02

+kt1     = -0.11       kt1l    = 0           kt2     = 0.022       ute     = -1.5
+ua1     = 4.31e-009   ub1     = 7.61e-018   uc1     = -5.6e-011   prt     = 0
+at      = 33000

+fnoimod = 1           tnoimod = 0

+jss     = 0.0001      jsws    = 1e-011      jswgs   = 1e-010      njs     = 1
+ijthsfwd= 0.01        ijthsrev= 0.001       bvs     = 10          xjbvs   = 1
+jsd     = 0.0001      jswd    = 1e-011      jswgd   = 1e-010      njd     = 1
+ijthdfwd= 0.01        ijthdrev= 0.001       bvd     = 10          xjbvd   = 1
+pbs     = 1           cjs     = 0.0005      mjs     = 0.5         pbsws   = 1
+cjsws   = 5e-010      mjsws   = 0.33        pbswgs  = 1           cjswgs  = 3e-010
+mjswgs  = 0.33        pbd     = 1           cjd     = 0.0005      mjd     = 0.5
+pbswd   = 1           cjswd   = 5e-010      mjswd   = 0.33        pbswgd  = 1
+cjswgd  = 5e-010      mjswgd  = 0.33        tpb     = 0.005       tcj     = 0.001
+tpbsw   = 0.005       tcjsw   = 0.001       tpbswg  = 0.005       tcjswg  = 0.001
+xtis    = 3           xtid    = 3

+dmcg    = 0e-006      dmci    = 0e-006      dmdg    = 0e-006      dmcgt   = 0e-007
+dwj     = 0.0e-008    xgw     = 0e-007      xgl     = 0e-008

+rshg    = 0.4         gbmin   = 1e-010      rbpb    = 5           rbpd    = 15
+rbps    = 15          rbdb    = 15          rbsb    = 15          ngcon   = 1
```

Liste des figures

Liste des tableaux

Glossaire

AADL	*Application & Architecture Design Language*
CMOS	*Complementary Metal Oxide Semiconductor*
DVS	*Dynamic Voltage Scaling*
ITRS	*International Technology Roadmap for Semiconductors*
LSB	*Least Significant Bits*
MDE	*Model Driven Engineering*
MOS	*Metal Oxide Semiconductor*
MPSoC	*Multi-Processors System on Chip*
MSB	*Most Significant Bits*
NoC	*Network on Chip*
OSATE	*Open Source AADL Tool Environment*
SoC	*System on Chip*
TDSI	*Traitement Du Signal et de l'Image*

Bibliographie

[AFP01] Y. Aghaghiri, F. Fallah, and M. Pedram. Irredundant address bus encoding for low power. In *the Proceedings of the 2001 international symposium on Low power electronics and design (ISLPED)*, pages 182–187, New York, NY, USA, 2001. ACM.

[BFB+04] B. Blampey, B. Fléchet, C. Bermond, V. Fontaine, G. Angénieux, J. Torres, and A. Farcy. Impact of copper dummies on interconnect propagation performance in advanced integrated circuits. *Microelectron. Eng.*, 76(1-4) :119–125, 2004.

[BM85] H.B. Bakaglu and J.D. Meindl. Optimal interconnection circuits for vlsi. *IEEE Trans on Electron. Devices*, 32(5) :903–909, 1985.

[BMM+97] L. Benini, G.D. Micheli, E. Macii, D. Sciuto, and C. Silvano. Asymptotic zero-transition activity encoding for address busses in low-power microprocessor-based systems. In *the Proceedings of the 7th Great Lakes Symposium on VLSI (GLS)*, page 77, Urbana, USA, 1997.

[BMM+98] L. Benini, G.D. Micheli, E. Macii, D. Sciuto, and C. Silvano. Address bus encoding techniques for system-level power optimization. In *the Proceedings of the conference on Design, automation and test in Europe (DATE)*, pages 861–867, Paris, France, 1998.

[CCH+06] G. Chen, H. Chen, M. Haurylau, N.A. Nelson, D.H. Albonesi, P.M. Fauchet, and E.G. Friedman. On-chip copper-based vs. optical interconnects : Delay uncertainty, latency, power, and bandwidth density comparative predictions. *International Interconnect Technology Conference*, pages 39–41, 2006.

[CF06] G. Chen and E.G. Friedman. Low-power repeaters driving rc and rlc interconnects with delay and bandwidth constraints. *IEEE Trans. on Very Large Scale Integration Systems*, Vol 14(No.2) :161–172, 2006.

[CJW+99] J.Y. Chen, W.B. Jone, J.S. Wang, H.I. Lu, and T.F. Chen. Segmented bus design for low-power systems. *IEEE Trans. on Very Large Scale Integration Systems*, 7(1) :25–29, 1999.

[CLJS07] A. Courtay, J. Laurent, N. Julien, and O. Sentieys. Estimation et optimisation de la consommation des interconnexions dans les soc. In *the 1st GDR SOC SIP symposium*, Paris, France, june 2007.

[CLJS08a] A. Courtay, J. Laurent, N. Julien, and O. Sentieys. Modélisation, estimation et optimisation de la consommation des interconnexions dans les soc. In *the 2nd GDR SOC SIP symposium*, Paris, France, june 2008.

[CLJS08b] A. Courtay, J. Laurent, N. Julien, and O. Sentieys. New directions in interconnect performance optimization. *3rd International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1–6, 2008.

[CLSJ07] A. Courtay, J. Laurent, O. Sentieys, and N. Julien. Modélisation et estimation de la consommation des interconnexions dans les soc. In *the Proceedings of the Faible Tension Faible Consommation conference (FTFC)*, pages 121–125, Paris, France, 2007.

[CLSJ08a] A. Courtay, J. Laurent, O. Sentieys, and N. Julien. Novel cross-transition elimination technique improving delay and power consumption for on-chip buses. In *the Proceedings of the International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages –, Lisbon, Portugal, 2008.

[CLSJ08b] A. Courtay, J. Laurent, O. Sentieys, and N. Julien. Procédé et dispositif de codage, système électronique et support d'enregistrement associés. Patent Pending, 2008. BFF 08P0103/HC.

[CSJ06] A. Courtay, O. Sentieys, and N. Julien. Interconnexions et consommation : où en sommes nous ? In *the Proceedings of the 4th MajecSTIC*, Lorient, France, 2006.

[CSLJ08a] A. Courtay, O. Sentieys, J. Laurent, and N. Julien. High-level interconnect delay and power estimation. *Journal of Low Power Electronics*, 4(1) :21–33, 2008.

[CSLJ08b] A. Courtay, O. Sentieys, J. Laurent, and N. Julien. Interconnect explorer : a high-level estimation tool for on-chip interconnects. In *the University Booth of the SAME Forum*, 2008.

[DP98] W.J. Dally and J.W. Poulton. *Digital Systems Engineering*. Cambridge University Press, 1998.

[DWM+05] W.R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A.M. Sule, M. Steer, and P.D. Franzon. Demystifying 3d ics : the pros and cons of going vertical. *Design and Test of Computers, IEEE*, 22(6) :498–510, 2005.

[Elm48] W.C. Elmore. The transient response of damped linear networks with particular regard to wideband amplifiers. *Journal of Applied Physics*, 19 :55–63, jan 1948.

[FPSS00] W. Fornaciari, M. Polentarutti, D. Sciuto, and C. Silvano. Power optimization of system-level address buses based on software profiling. In *the Proceedings of the 8th international workshop on Hardware/software codesign (CODES)*, pages 29–33, San Diego, USA, 2000.

[Gok05] H.S. Gokturk. Electrical properties of ideal carbon nanotubes. *5th IEEE Conference on Nanotechnology*, 2 :677–680, 2005.

[GV98] T. Givargis and F. Vahid. Interface exploration for reduced power in core-based systems. In *the Proceedings of the 11th international symposium on System synthesis (ISSS)*, pages 117–122, Washington, DC, USA, 1998. IEEE Computer Society.

[HMH01] R. Ho, K. Mai, and M. Horowitz. The future of wires. *Proceedings . IEEE*, 89(4) :490–504, April 2001.

[HY00] K. Hirose and H. Yasuura. A bus delay reduction technique considering crosstalk. In *the Proceedings of the conference on Design, automation and test in Europe (DATE)*, pages 441–445, Paris, France, 2000.

[IFN99] Y.I. Ismail, E.G. Friedman, and J.L. Neves. Figures of merit to characterize the importance of on-chip inductance. *IEEE Trans. on Very Large Scale Integration Systems*, 7(4) :442–449, 1999.

[ITR02] ITRS. Technical report. *International Technology Roadmap for Semiconductors*, 2002.

[ITR04] ITRS. Technical report. *International Technology Roadmap for Semiconductors*, 2004.

[ITR06] ITRS. Technical report. *International Technology Roadmap for Semiconductors*, 2006.

[KBSV78] S.P. Khatri, R.K. Brayton, and A.L. Sangiovanni-Vincentelli. *Crosstalk Noise Immune VLSI Design Regular Layout Fabrics*. Kluwer Academic Publishers, 1978.

[KIA99] S. Komatsu, M. Ikeda, and K. Asada. Low power chip interface based on bus data encoding with adaptive code-book method. In *the Proceedings of the 9th IEEE Great Lakes Symposium on VLSI (GLS)*, pages 368–371, Ann Arbor, USA, 1999.

[KKI+05] A. Kurokawa, T. Kanamoto, T. Ibe, A. Kasebe, C.W. Fong, T. Kage, Y. Inoue, and H. Masuda. Dummy filling methods for reducing interconnect capacitance and number of fills. In *the Proceedings of the 6th International Symposium on Quality of Electronic Design (ISQED)*, pages 586–591, Washington, DC, USA, 2005. IEEE Computer Society.

[KNM04] C. Kretzschmar, A.K. Nieuwland, and D. Muller. Why transition coding for power minimization of on-chip buses does not work. In *the Proceedings of the conference on Design, automation and test in Europe (DATE)*, pages 10512–10517, Paris, France, 2004.

[Kob05] M.J. Kobrinsky. On-chip optical interconnects. *Intel Technology Journal*, 8(2) :129–141, 2005.

[LCL95] J. Lillis, C.K. Cheng, and T.T.Y. Lin. Optimal wire sizing and buffer insertion for low power and a generalized delay model. In *the Proceedings of the 1995 IEEE/ACM international conference on Computer-aided design (ICCAD)*, pages 138–143, Washington, DC, USA, 1995. IEEE Computer Society.

[LHLW03] T. Lv, J. Henkel, H. Lekatsas, and W. Wolf. A dictionary-based en/decoding scheme for low-power data buses. *IEEE Trans. on Very Large Scale Integration Systems*, 11(5) :943–951, 2003.

[Lie03] L.W. Liebmann. Layout impact of resolution enhancement techniques : impediment or opportunity ? In *the Proceedings of the 2003 international symposium on Physical design (ISPD)*, pages 110–117, New York, NY, USA, 2003. ACM.

[LL75] H.C. Lin and L.W. Linholm. An optimized output stage for mos integrated circuits. *IEEE Journal of Solid-State Circuits*, 10 :106–109, April 1975.

[LMHY05] X.C. Li, J.F. M, H.F. Huang, and Y.Liu. Global interconnect width and spacing optimization for latency, bandwidth and power dissipation. *IEEE Trans on Electron. Devices*, Vol 52(No.10) :2272–2279, 2005.

[LR95] P.E. Landman and J.M. Rabaey. Architectural power analysis : the dual bit type method. *IEEE Trans. on Very Large Scale Integration Systems*, 3(2) :173–187, 1995.

[MC80] C. Mead and L. Conway. *Introduction to VLSI Systems.* Addison-Wesley, 1980.

[MKWS04] N. Magen, A. Kolodny, U. Weiser, and N. Shamir. Interconnect-power dissipation in a microprocessor. In *the Proceedings of the 2004 international workshop on System level interconnect prediction (SLIP)*, pages 7–13, Paris, France, 2004.

[MMP02] L. Macchiarulo, E. Macii, and M. Poncino. Wire placement for crosstalk energy minimization in address buses. In *the Proceedings of the conference on Design, automation and test in Europe (DATE)*, pages 158–162, Paris, France, 2002.

[Moo65] G.E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8) :114–117, 1965.

[NB00] A. Nalamalpu and W.P. Burleson. Repeater insertion in deep sub-micron cmos : ramp-based analytical model and placement sensitivity analysis. In *the Proceedings of the International Symposium on Circuits and Systems*, 2000.

[NB01] A. Nalamalpu and W.P. Burleson. Optimal wire sizing and buffer insertion for low power and a generalized delay model. In *the Proceedings of the IEEE international conference on ASIC/SOC*, 2001.

[Nem84] N. Nemes. Driving large capacitance in mos lsi systems. *IEEE Trans. of Solid-State Circuits*, 19(1) :159–161, 1984.

[NM05] A. Naeemi and J.D. Meindl. Monolayer metallic nanotube interconnects : promising candidates for short local interconnects. *IEEE Electron Device Letters*, 26(8) :544–546, 2005.

[PPS06] J.M. Philippe, S. Pillement, and O. Sentieys. Area efficient temporal coding schemes reducing crosstalk effects. In *the Proceedings of the International Symposium on Quality Electronic Design (ISQED)*, pages 334–339, San Jose, USA, 2006.

[PTM07] PTM. Interconnect. *Predictive Technology Model*, 2007.

[RCN03] J.M. Rabaey, A. Chandrakasan, and B. Nikolic. *Digital Integrated Circuits : A design perspective.* Pearson Education, 2003.

[Sak93] T. Sakurai. Closed-form expressions for interconnection delay, coupling, and crosstalk in vlsi's. In *IEEE Trans on Electron. Devices*, volume 40, pages 118–124, 1993.

[SB95] M.R. Stan and W.P. Burleson. Bus-invert coding for low-power i/o. In *IEEE Trans. on Very Large Scale Integration Systems*, volume Vol 3, pages pp.49–58, 1995.

[SCC98] Y. Shin, S.-IK Chae, and K. Choi. Partial bus-invert coding for power optimization of system level bus. In *the Proceedings of the 1998 international symposium on Low power electronics and design (ISLPED)*, pages 127–129, New York, NY, USA, 1998. ACM Press.

[SD95] C.L. Su and A.M. Despain. Cache design trade-offs for power and performance optimization : a case study. In *the Proceedings of the international symposium on Low power design (ISLPED)*, pages 63–68, Dana Point, USA, 1995.

[SPJ03] L. Shang, L. Peh, and N.K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. In *the Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA)*, pages 91–102, Anaheim, USA, 2003.

[STD94] C.L. Su, C.Y. Tsu, and A.M. Despain. Saving power in the control path of embedded processors. *Design and Test of Computers, IEEE*, 11 :24–31, 1994.

[TDZ01] C.N. Taylor, S. Dey, and Y. Zhao. Modeling and minimization of interconnect energy dissipation in nanometer technologies. In *the Proceedings of the 38th conference on Design automation (DAC)*, pages 754–757, Las Vegas, USA, 2001.

[The00] T.N. Theis. The future of interconnction technology. *IBM J. Research and Development*, 44(3), may 2000.

Bibliographie Personnelle

Brevet

[CLSJ08a] A. Courtay and J. Laurent and O. Sentieys and N. Julien. Procédé et dispositif de codage, système électronique et support d'enregistrement associés. *Patent Pending*, Reference BFF 08P0103/HC, 2008

Revue internationale

[CSLJ08b] A. Courtay and O. Sentieys and J. Laurent and N. Julien. High-Level Interconnect Delay and Power Estimation. *Journal of Low Power Electronics (JOLPE)*, Vol 4, No 1, pages 21-33, 2008

Conférences internationales avec comité de lecture

[CLSJ08c] A. Courtay and J. Laurent and O. Sentieys and N. Julien. Novel Cross-Transition Elimination Technique Improving Delay and Power Consumption for On-Chip Buses. In *the Proceedings of the International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages - , Lisbon, Portugal, 2008

[CLJS08d] A. Courtay and J. Laurent and N. Julien and O. Sentieys. New directions in interconnect performance optimization. In *the Proceedings of the 3rd IEEE International Conference on Design and Technology of Integrated Systems in Nanoscale Era (DTIS)*, pages 1-6, Tozeur, Tunisia, 2008

Conférences nationales avec comité de lecture

[CLSJ07a] A. Courtay and J. Laurent and O. Sentieys and N. Julien. Modélisation et estimation de la consommation des interconnexions dans les SOC. In *the Proceedings of the Faible Tension Faible Consommation conference (FTFC)*, pages 121-125, Paris, France, 2007

[CSJ06] A. Courtay and O. Sentieys and N. Julien. Interconnexions et consommation : où en sommes nous ? In *the 4th MajecSTIC conference (MajecSTIC)*, Lorient, France, 2006

Conférences nationales sans actes

[CLJS08f] A. Courtay and J. Laurent and N. Julien and O. Sentieys. Modélisation, Estimation et Optimisation de la consommation des interconnexions dans les SOC. In *the 2nd GDR SOC SIP symposium*, Paris, France, 2008

[CLSJ07b] A. Courtay and J. Laurent and N. Julien and O. Sentieys. Estimation et optimisation de la consommation des interconnexions dans les SOC. In *the 1st GDR SOC SIP symposium*, Paris, France, 2007

University Booth

[CSLJ08e] A. Courtay and O. Sentieys and J. Laurent and N. Julien. Interconnect Explorer : a High-Level Estimation Tool for On-Chip Interconnects. In *the University Booth of the SAME Forum (SAME)*, Nice, France, 2008

Abstract

Nowadays, nomad applications are more and more complex and require many computationnal ressources, which involve a large amount of data to be stored or translated from a unit to another. Moreover, with technological parameters evolution, controlling propagation time and power consumption of SoC's interconnects becomes a major issue. ITRS's predictions show wire and transistor dimensions shrinking, which imply a circuit behaviour modification ; especially with propagation time. Today the wire propagation time becomes higher than the gate one. This increase is among other things, due to the increase of interconnect's resistance and capacitance. The capacitance increase also involves a power consumption increase due to interconnects which can represents up to 50% of the total chip power consumption and area.
So it is now necessary to take interconnect's power consumption into acount during the chip power consumption evaluation. To do this, accurate physical interconnects models and power consumption estimation tools have to be proposed to enable designers having reliable results on the chip design.

In the first chapter of this thesis, physical bus modeling for power consumption modeling is discussed. Distributed resistance and capacitance wire has first be caracterized, then for buses, buffers and crosstalk capacitances have been considered.

In the second chapter, the interconnect power consumption estimation methodology is disscussed. As the bus has been physically modeled, important parameters that impact power consumption (technology, metal layer, bus length ...) have been extracted. Finally, SPICE simulations of the circuits have been done ; experimental results provided by the simulations have allowed us to realise some models which have been included in our estimation tool. Our tool (*Interconnect Explorer*) allow users, after configuration (which means choosing a technology, a metal layer, a bus length and so on) to obtain rapidely a power consumption estimation of the considered bus. Validation experimentations show that the maximum error of the estimation tool is 3% (compared to SPICE simulations) with a few seconds execution time (a SPICE simulation in the same experimental conditions can last few hours).

In the third chapter, a state of the art of the major power and timing optimization techniques is proposed. *Interconnect Explorer* allows us to validate the techniques efficiency on the power consumption impacting parameters (activity, propagation time, parasitic capacitances ...). Then, the analysis of the results provided by *Interconnect Explorer* allows us to demonstrate that optimization techniques do not face all good criterias. At the chapter end, some new ways for interconnect power consumption optimization are proposed.

The fourth chapter of this thesis presents our power consumption optimization techniques according to the issues disscussed in the previous chapter. The particularity of these techniques (one of them called the *Spatial Switching* is patented) is that they have a low material overhead. Many methodologies proposed in the state of the art have a quite high material overhead, particularly due to their codecs. These codecs lead to a power consumption overhead often higher than the power consumption reduction they can lead on the bus for usual *SoC* interconnect length. Our *Spatial Switching* experimental results show energetic power consumption gains that can rise up to 12% for a 5mm bus in the 65nm technology. These results include, of course, the extra power consumption due to the codecs. Gains rise more with technological steps and bus length increase.

We will also propose a possible extension of our work (tool and models) by the abstracton level elevation. In our work, point to point interconnects have been considered ; but, present systems can use more complex communication schemes. First, our approach can be used to model *MESH* or *NoC* interconnects that are often use in *MPSoC* systems. Experimental results will be extracted from the simulation of *MPSoC* architectures using the *SocLib* platform. Then, these results can be extended to be used in a *MDE* (*Model Driven Engineering*) approach. In this context, our work will be included in the *ITEA SPICES* project using an *AADL* profile (*Application & Architecture Design Language*). The goal is, here, to use our results in the *OSATE* (*Open Source AADL Tool Environment*) framework to allow the power consumption estimations during the first design phases of the system.
As interconnect power consumption has become a major issue in *SoC* design, this thesis will be concluded by a presentation of the emerging interconnect design solutions (optical interconnects, 3D *SoC*, carbon nanotubes...) and how our work can be applied on these technologies.

Keywords : Interconnects, bus, modeling, estimating, architectural optimization, power consumption, coding

Résumé

Aujourd'hui les applications portables sont de plus en plus complexes et nécessitent beaucoup de ressources de calculs, ce qui implique un fort volume de données à stocker ou à faire transiter d'une unité à une autre. De plus, avec l'évolution des paramètres technologiques, la maîtrise de l'évolution du délai et de la consommation des interconnexions au sein d'un *SoC* (*System On Chip*) est de plus en plus difficile à contrôler. Les prévisions de l'ITRS montrent une diminution des dimensions des transistors et des fils, ce qui se traduit par une évolution du comportement du circuit tout particulièrement au niveau temporel. Ainsi, le délai d'un fil devient largement supérieur à celui d'une porte. Cette augmentation est due à l'évolution des paramètres résistifs et capacitifs des interconnexions qui tendent toujours à augmenter. L'augmentation des phénomènes capacitifs se traduit également par une augmentation de la part de la consommation due aux interconnexions qui peut représenter jusqu'à 50% de la consommation totale et de la surface occupée sur la puce. Il devient donc indispensable de prendre en compte les interconnexions lors de l'évaluation de la consommation d'une puce. Pour cela, des modèles précis des interconnexions doivent être proposés ainsi que des outils d'estimation afin de fournir aux concepteurs des retours rapides et fiables sur leur *design*. Des techniques d'optimisation doivent également être proposées et leur impact quantifié par le biais entre autre des outils d'estimation.

Le premier chapitre de la thèse se propose, d'aborder la modélisation de la consommation d'un bus à l'aide de modèles physiques des différents éléments entrant dans sa composition. Le fil sous forme de modèles résistifs et capacitifs distribués a d'abord été caractérisé, puis, au niveau bus, nous avons caractérisé les buffers ainsi que les diaphonies capacitives entre fils.

Dans le second chapitre, la méthode d'estimation de la consommation des interconnexions est proposée. Suite à la modélisation du bus au niveau technologique, les paramètres importants intervenant dans la variation de la consommation (technologie, couche de métal, longueur de bus...) ont été extraits. Des simulations SPICE de ces circuits ont été réalisées ; les résultats expérimentaux ont permis d'obtenir des modèles inclus au sein d'un outil d'estimation. Cet outil (*Interconnect Explorer*) permet alors à l'utilisateur, après configuration, (c'est-à-dire choix de la technologie, de la couche de métal, de la longueur de bus) d'obtenir très rapidement une estimation de la consommation du transfert de données sur un bus. Les expérimentations de validation montrent que l'outil permet d'obtenir une estimation avec une erreur maximale de 3% (par rapport aux simulations SPICE) avec un temps d'exécution de quelques secondes (une simulation SPICE dans les mêmes conditions expérimentales prenant plusieurs heures).

Dans le troisième chapitre, un état de l'art des principales techniques d'optimisation de la consommation et du délai est présenté. L'outil d'estimation présenté dans le chapitre précédent nous permet de valider l'efficacité de ces techniques sur les paramètres impactant la consommation (activité, temps de propagation, capacités parasites...). Dans un second temps, l'analyse des résultats fournis par l'outil permet de montrer que les techniques d'optimisation n'agissent pas forcément sur les bons paramètres. A la fin de ce chapitre, de nouvelles pistes d'optimisation, en adéquation avec les résultats précédents, sont proposées.

Le quatrième chapitre présente les techniques d'optimisation au niveau architectural auxquelles nous avons abouti en se basant sur les pistes d'optimisation du chapitre précédent. Ces techniques (dont une est brevetée : *Spatial Switching*) ont pour particularité de nécessiter un surcoût matériel relativement faible. En effet, nombre des méthodes présentées dans la littérature ont un surcoût matériel assez important, en particulier dû aux codeurs et décodeurs. Ces codecs engendrent un surcoût en consommation bien souvent supérieur à la réduction apportée sur le bus pour des longueurs d'interconnexions usuelles dans les *SoC* actuels. Nos résultats expérimentaux sur le *Spatial Switching* montrent des gains en consommation pouvant atteindre une réduction de 12% de consommation d'énergie pour un bus de 5mm en 65nm. Ces résultats incluent bien évidemment la consommation due aux codecs. Les gains augmentent encore avec les sauts technologiques ainsi qu'avec l'augmentation de la longueur du bus.

Nous proposerons également une extension possible de nos travaux (outil et modèles) par l'élévation du niveau d'abstraction. En effet, dans ce mémoire, les interconnexions point à point sont notre principale préoccupation ; or, les systèmes actuels peuvent utiliser des réseaux de communication plus complexes. Dans un premier temps, notre approche peut être utilisée pour modéliser des interconnexions de type *MESH* ou *NoC* souvent utilisées dans le cadre de systèmes *MPSoC* (utilisation des résultats de la plate forme SocLib). Dans un second temps, ces résultats et les précédents peuvent être étendus afin d'être utilisés dans une approche *MDE* (*Model Driven Engineering*). Dans ce cadre, nos travaux s'intégreront dans le projet *ITEA SPICES* qui utilise un profil *AADL* (*Application & Architecture Design Language*), le but étant, ici, d'intégrer nos résultats dans le "framework" *OSATE* afin de pouvoir estimer la consommation des communications dès les premières phases de conception.
La consommation des interconnexions étant devenu un enjeu majeur dans la conception de système, nous conclurons la thèse par une présentation des futures technologies d'interconnexions alternatives à la conception classique : interconnexions optiques, *SoC* 3D, nanotubes...

Mots-clés : Interconnexion, bus, modélisation, estimation, optimisation architecturale, consommation, codage